









obj -> BinRename(alpha1) : int        

Verzeichnis/Objekt umbenennen

obj Objekt/Verzeichnis

alpha1 Neuer Name

Umbenennungsergebnis

ErrOk Verschieben

erfolgreich

ErrBinNoLock Objekt/Verzeichnis

(obj) nicht

Resultat int gesperrt  ErrBinExists Objekt/Verzeichnis

mit Namen(alpha)

existiert bereits

rDeadlock Verklemmung

aufgetreten

Siehe Verwandte Befehle, BinCopy(),

BinMove()

Mit dieser Funktion wird das Objekt/Verzeichnis (obj) nach (alpha1) umbenannt. Das Objekt/Verzeichnis muss dazu exklusiv gesperrt sein (siehe BinLock und BinSingleLock).

Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt/Verzeichnis (obj) ungültig

## Kontakt

obj -> BinWriteMem(handle1[, int2[, alpha3]]) : int



### Binäres Objekt aus Memory-Objekt schreiben

obj  
Deskriptor eines  
binären Objekts  
handle1 Deskriptor eines  
Memory-Objekts  
Kompressionsstufe  
int2  
(optional)  
Verschlüsselungs-Code

alpha3 (optional) —

Siehe Verwandte Befehle,

BinReadMem()

Mit dieser Funktion wird der komplette Inhalt des Memory-Objekts (handle1) in das binäre Objekt (obj) geschrieben. Ein bereits bestehender Inhalt wird dabei überschrieben. Das Objekt muss dazu exklusiv gesperrt sein (siehe BinLock oder BinSingleLock).

Optional kann der Inhalt durch Übergabe einer der Stufen 1 bis 4 in (int2) komprimiert werden. Eine Kompressionsstufe sollte nicht bei Dateien angegeben werden, die sich nicht weiter komprimieren lassen. Dazu gehören vor allem gepackte Dateiformate (.zip, .rar usw.) und komprimierte Multimedia-Formate (.jpg, .mov, .mp3 usw.).

Optional kann das Objekt mit einer symmetrischen Verschlüsselung gespeichert werden. Dazu wird ein entsprechender Verschlüsselungscode mit bis zu 64 Zeichen in (alpha2) übergeben (siehe StrEncrypt()). Es ist zu beachten, dass ohne diesen Code der Objektinhalt nicht mehr gelesen werden kann.

Das Resultat ist ErrOk, wenn die Daten korrekt geschrieben werden konnten. Es können folgende Fehlerresultate auftreten:

ErrBinNoLock Das binäre Objekt ist nicht exklusiv gesperrt.

ErrBinNoData Das Memory-Objekt enthält keine Daten rDeadlock  
Verklemmung aufgetreten

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder (handle1) angegebene Desriptor ist ungültig.

## Kontakt

obj -> BinUpdate() : int        Änderungen an \_\_\_\_\_

Verzeichnis/Objekt übernehmen

obj      Objekt/Verzeichnis

    Übernahmeresultat

ErrOk

    Übernehmen erfolgreich

Resultat int ErrBinNoLock Objekt/Verzeichnis (obj) nicht



    gesperrt

rDeadlock

    Verklemmung aufgetreten

Siehe Verwandte Befehle

Mit diesem Befehl werden Änderungen an den Eigenschaften eines Objekts oder eines Verzeichnisses in der Datenbank gespeichert. Das Objekt/Verzeichnis muss dazu exklusiv gesperrt sein (siehe BinLock und BinSingleLock).

Veränderte Eigenschaften werden auch durch die Befehle BinRename(), BinCopy(), BinMove() und BinImport() gespeichert.

Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt/Verzeichnis (obj) ungültig

Konstanten für binäre Objekte

Konstanten für binäre Objekte

Befehle  
für

Siehe

binäre

- BinClearOnly
- BinCreate
- BinCreateNew
- BinDbg?
- BinDeleteAll
- BinDirectory
- BinErrorDecryption
- BinFirst
- BinLast
- BinLock
- BinNext
- BinPrev
- BinSharedLock
- BinSingleLock

**\_BinClearOnly**

**Objektinhalt löschen**

Wert 32.768 /

**0x00008000**

**Verwandte**

Siehe **Befehle**,

**BinDelete()**

Option bei **BinDelete()** durch die der Inhalt eines binären Objekts gelöscht werden kann.

\_BinCreate

Binäres Objekt oder Verzeichnis erstellen

Wert 4.096 /

**0x00001000**

Verwandte

Befehle,

Siehe BinOpen(),

BinDirOpen(),

BinCreateNew

Option bei BinOpen() und BinDirOpen() durch die ein neues binäres Objekt oder Verzeichnis erstellt werden kann.



Der Name darf nicht auf ein Backslash (\) enden.

\_BinCreateNew

Neues binäres Verzeichnis oder Objekt explizit anlegen

Wert 36.864 /

**0x00009000**

Verwandte

Befehle,

Siehe BinOpen(),

BinDirOpen(),

BinCreate

Option bei BinOpen() und BinDirOpen() durch die ein neues binäres Objekt oder Verzeichnis explizit erstellt werden kann. Existiert das Verzeichnis oder Objekt bereits, wird der Fehlerwert ErrBinExists zurückgegeben.



Der Name darf nicht auf ein Backslash (\) enden.

## Kontakt

\_BinDba?

Objekt/Verzeichnis in anderer Datenbank ansprechen Wert

0x10000  
0x70000 -

### Verwandte

Siehe Befehle,

### DbaConnect()

Option bei BinOpen(), BinDirOpen(), BinDelete() und BinDirDelete() durch die ein binäres Objekt/Verzeichnis in einer anderen Datenbank angesprochen werden kann.

Zuvor muss diese Datenbank mit dem Befehl DbaConnect() verbunden werden. Der dabei angegebene Nummernbereich bestimmt, mit welcher Option binäre Objekte/Verzeichnisse dieser Datenbank angesprochen werden können:

Nummernbereich 2 : \_BinDba2

Nummernbereich 3 : \_BinDba3

Nummernbereich 4 : \_BinDba4

Nummernbereich 5 : \_BinDba5

Nummernbereich 6 : \_BinDba6

Nummernbereich 7 : \_BinDba7

Nummernbereich 8 : \_BinDba8



**\_BinDeleteAll**

**Verzeichnis leeren und löschen**

Wert 16.384 /

**0x00004000**

**Verwandte**

Siehe **Befehle**,

**BinDirDelete()**

Option bei **BinDirDelete()** durch die ein Verzeichnis mit allen Unterverzeichnissen und Objekten gelöscht werden kann.

**\_BinDirectory**

**Unterverzeichnis lesen**

**Wert 8.192 /**

**0x00002000**

**Verwandte**

**Siehe Befehle,**

**BinDirRead()**

**Option bei BinDirRead() durch die alle Unterverzeichnisse eines Verzeichnisses gelesen werden können.**

## Kontakt

### \_BinErrorDecryption

Eindeutiger Fehler zurückgeben, wenn Entschlüsselungscode falsch Wert 1 / 0x00000001

#### Verwandte

Siehe Befehle,

BinExport(),

BinReadMem()

Option bei BinExport() oder BinReadMem() durch die bei einem falschen Entschlüsselungscode der Fehler \_ErrBinDecryption statt dem allgemeinen Fehler \_ErrBinData zurückgegeben werden kann.

## Kontakt

**\_BinFirst**

**Ersten Eintrag lesen**

Wert <sup>1/</sup>

**0x00000001**

**Verwandte**

Siehe **Befehle**,

**BinDirRead()**

Option bei **BinDirRead()** durch die der erste Eintrag in einem Verzeichnis gelesen werden kann.

## Kontakt

**\_BinLast**

**Letzten Eintrag lesen**

Wert<sup>2/</sup>

**0x00000002**

**Verwandte**

Siehe **Befehle**,

**BinDirRead()**

Option bei **BinDirRead()** durch die der letzte Eintrag in einem Verzeichnis gelesen werden kann.

**\_BinLock**

**Objekt/Verzeichnis für andere Benutzer sperren**

Wert

**0x00000008**

**Verwandte**

Siehe **Befehle**,

**BinOpen()**,

**BinDirOpen()**

**Option bei BinOpen() und BinDirOpen() durch die ein Objekt/Verzeichnis beim Öffnen/Anlegen für andere Benutzer gesperrt werden kann.**

**\_BinNext**

**Eintrag nach Referenzeintrag lesen**

Wert <sup>4</sup>/<sub>P</sub>

**0x00000004**

**Verwandte**

Siehe **Befehle**,

**BinDirRead()**

Option bei **BinDirRead()** durch die der Eintrag nach dem Referenzeintrag in einem Verzeichnis gelesen werden kann.

## Kontakt

\_BinPrev

Eintrag vor Referenzeintrag lesen

Wert <sup>3</sup>/<sub>P</sub>

0x00000003

Verwandte

Siehe Befehle,

BinDirRead()

Option bei BinDirRead() durch die der Eintrag vor dem Referenzeintrag in einem Verzeichnis gelesen werden kann.



## Kontakt

**\_BinSharedLock**

**Objekt/Verzeichnis gemeinsam mit anderen Benutzer sperren**

Wert <sup>48</sup>/

**0x00000030**

**Verwandte**

Siehe Befehle,

**BinOpen()**,

**BinDirOpen()**

**Option bei BinOpen() und BinDirOpen() durch die ein Objekt/Verzeichnis beim Öffnen/Anlegen gemeinsam mit anderen Benutzer gesperrt werden kann.**

**\_BinSingleLock**

**Objekt/Verzeichnis für alle Benutzer sperren**

Wert <sup>40</sup> /

**0x00000028**

**Verwandte**

Siehe **Befehle**,

**BinOpen()**,

**BinDirOpen()**

**Option bei BinOpen() und BinDirOpen() durch die ein Objekt/Verzeichnis beim Öffnen/Anlegen für alle Benutzer gesperrt werden kann.**

## Kontakt

### Befehle für Storage-Objekte

Liste der Befehle und Konstanten zur Bearbeitung von Storage-Objekten und -Verzeichnissen

Befehlsgruppen,

Siehe Befehlsliste,

Storage-Objekte

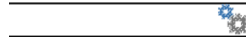
#### Befehle

- StoClose
- StoDelete
- StoDirOpen
- StoDirRead
- StoExport
- StoImport
- StoImportTile
- StoOpen
- StoReadMem
- StoWriteMem
- StoWriteTileMem

#### Konstanten

- StoCreate
- StoDb2
- StoDb3
- StoDb4
- StoDb5
- StoDb6
- StoDb7
- StoDb8
- StoDirectory
- StoFirst
- StoLast
- StoNext
- StoPrev

obj -> StoClose()



**Storage-Objekt schließen**

obj      Deskriptor des


**Storage-Objekts**

**Verwandte**

Siehe Befehle,

StoOpen(),

StoDirOpen()

Das in (obj) übergebene Storage-Objekt  das zuvor mit StoOpen() geöffnet wurde, wird geschlossen. Der Deskriptor ist anschließend nicht mehr gültig.

**Mögliche Laufzeitfehler:**

**ErrHdlInvalid** Der übergebene Deskriptor ist ungültig.

## Kontakt

obj -> StoDelete(alpha1[,  
int2]) : int



Storage-Objekt löschen

obj Ausgangsverzeichnis

alpha1 Objektname

int2 Optionen (optional)

StoDb? Datenbankbereich

Löschresultat

ErrOk

Löschen

erfolgreich

ErrRights

Keine

ausreichenden

Rechte

ErrStoNameInvalid Objektname

(alpha1)

ungültig

ErrStoNoFile

Objekt in

Objektname

(alpha1)



existiert nicht

ErrStoNoPath

Verzeichnis in

Objektname

(alpha1)

existiert nicht

ErrStoLocked

Objekt

(alpha1)

gesperrt

\_rDeadlock

Verklemmung

aufgetreten

Resultat int

Siehe Verwandte Befehle, StoOpen()

Mit dieser Funktion wird ein Storage-Objekt gelöscht. In (obj) wird der Deskriptor des Ausgangsverzeichnisses angegeben.



Das Ausgangsverzeichnis darf nicht das Wurzelverzeichnis sein. Es muss vorher ein Pfad mit StoDirOpen() geöffnet werden.

Folgende Optionen (int2) können angegeben werden:

- StoDb?

Das Objekt wird in einer mit DbConnect() verbundenen Datenbank gelöscht oder geleert.

Der Datenbankbereich wird in der Option mit StoDb2 bis StoDb8 angegeben.

Beispiele:

```
// Objekt 'File' im Verzeichnis tHdl löschentHdl->StoDelete('File');// Objekt 'File' in verbunden
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Ausgangsverzeichnis (obj) ungültig

obj -> StoDirRead(int1[, alpha2]) : alpha               Verzeichniseintrag  
eines Storage-Verzeichnisses lesen

obj Eltern-Verzeichnis

int1 Optionen

alpha2 Name des Verzeichniseintrags  
(optional)

Resultat alpha Name des Verzeichniseintrags

Siehe Verwandte Befehle, StoDirOpen()

Mit dieser Anweisung wird ein Verzeichniseintrag aus dem übergebenen Verzeichnis gelesen. Das Verzeichnis muss zuvor mit der Anweisung StoDirOpen() geöffnet worden sein. Der von dieser Anweisung zurückgegebene Deskriptor wird in (obj) übergeben. Über die Optionen in (int1) kann bestimmt werden, welcher Eintrag gelesen werden soll. Folgende Optionen können angegeben werden:

0 Keine Option angegeben. Es wird der Verzeichniseintrag, der in (alpha2) angegeben ist gelesen.

StoFirst Der erste Verzeichniseintrag wird gelesen.

StoPrev Der vorhergehende Verzeichniseintrag wird gelesen. In (alpha2) muss ein Referenz-Objekt angegeben werden.

StoNext Der nächste Verzeichniseintrag wird gelesen. In (alpha2) muss ein Referenz-Objekt angegeben werden.

StoLast Der letzte Verzeichniseintrag wird gelesen.

Wird als Option 0 angegeben muss in (alpha2) ein Verzeichniseintrag übergeben werden. Bei den Optionen StoPrev und StoNext kann ein Verzeichniseintrag übergeben werden, um ab einem bestimmten Eintrag zu lesen.

Kann kein Verzeichniseintrag gelesen werden, wird eine leere Zeichenkette zurückgegeben.

Beispiel:

```
tHdlStoDir # StoDirOpen(0, 'PrintForm');for tName # tHdlStoDir->StoDirRead(_StoFirst);loop tNa
```

## Kontakt

obj -> StoDirOpen(alpha1[, int2]) :



handle

Öffnen eines Storage-Verzeichnisses

obj Eltern-Verzeichnis

alpha1 Name des Verzeichnis

Optionen (optional)

int2 StoDba? Lesen aus verbundener

Datenbank

Deskriptor des

Resultat handle Storage-Verzeichnisses oder

Fehlerwert

Siehe Verwandte Befehle, StoOpen(), StoClose()

Diese Anweisung öffnet ein Storage-Verzeichnis und gibt den Deskriptor des Verzeichnisses zurück.

In (obj) wird das übergeordnete Verzeichnis übergeben. 0 entspricht dabei dem Root-Verzeichnis.

Als Verzeichnisse können folgende Namen in (alpha1) übergeben werden:

'Dialog'	Verzeichnis der Dialog-Objekte
'Menu'	Verzeichnis der Menü-Objekte
'PrintForm'	Verzeichnis der PrintForms
'PrintFormList'	Verzeichnis der Drucklisten
'PrintDocument'	Verzeichnis der Druckdokumente
'PrintDocTable'	Verzeichnis der Drucktabellen
'Picture'	Verzeichnis der Raster- und Kachelgrafiken
'MetaPicture'	Verzeichnis der Vektorgrafiken
'UITheme'	Verzeichnis der <u>Themes</u>

'Preview\Dialog' Verzeichnis der Vorschaubilder von Dialog-Objekten

Als optionaler Parameter (int2) kann eine der Konstanten StoDba2 bis StoDba8 angegeben werden, wenn Storage-Verzeichnisse aus einer mit DbaConnect() verbundenen Datenbank geöffnet werden sollen.

Über den zurückgegebenen Deskriptor können die Eigenschaften (ID, Name, FullName und Custom) gelesen werden.

Beispiel:

```
tHdlStoDir # StoDirOpen(0, 'Dialog');if (tHdlStoDir > 0){ ... tHdlStoDir->StoClose();}
```

Folgende Fehlerwerte werden durch den Befehl zurückgegeben:

ErrRights Keine ausreichenden Rechte

ErrStoNameInvalid In (alpha1) wurde ein ungültiger Verzeichnisname angegeben.

ErrStoNoPath Der in (alpha1) angegebene Verzeichnisname existiert nicht.



## Kontakt

obj ->

StoExport(alpha1) 

: int

Objekt exportieren

obj      Objekt

alpha1      Name      Datei      der externen

Exportresultat



Resultat int (siehe Text)

Verwandte Befehle,

Siehe StoReadMem(),

StoOpen()

Mit dieser Funktion wird der Inhalt des Storage-Objektes (obj) in die externe Datei (alpha1) exportiert.

Die Namen der in der Datenbank enthaltenen Objekte können mit dem Befehl StoDirRead() ermittelt werden.

Beim Export des Storage-Objektes wird das Originaldatum und die Originalzeit der Datei wieder hergestellt.



Das Exportieren von Resource-Elementen wie Dialogen und Menüs ist derzeit nicht möglich.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:

<u>ErrRights</u>	Keine ausreichenden Rechte
<u>ErrStoNoData</u>	Das Storage-Objekt enthält keine Daten
<u>ErrFsiNoPath</u>	Externer Pfad nicht vorhanden
<u>ErrFsiOpenOverflow</u>	Maximale Anzahl offener Dateien erreicht
<u>ErrFsiAccessDenied</u>	Zugriff auf externe Datei (alpha1) verweigert
<u>ErrFsiHdlInvalid</u>	Datei-Deskriptor von externer Datei (alpha1) ungültig
<u>ErrFsiDriveInvalid</u>	Laufwerk im Namen der externen Datei (alpha1) ungültig
<u>ErrFsiSharingViolation</u>	Zugriffskonflikt bei Zugriff auf externe Datei (alpha1)
<u>ErrFsiLockViolation</u>	Sperrkonflikt bei Zugriff auf externe Datei (alpha1)
<u>ErrFsiOpenFailed</u>	Externe Datei (alpha1) konnte nicht geöffnet werden

Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt (obj) ungültig

## Kontakt

obj -> StoImport(alpha1) :



int

Storage-Objekt importieren

obj        Objekt

alpha1    Pfad und Name einer  
          externen Datei

Resultat int Fehlerwert (siehe Text)  —

Verwandte Befehle,

Siehe      StoWriteMem(), StoExport(),  
            StoOpen()

Mit dieser Funktion wird der Inhalt der externen Datei (alpha1) in das Storage-Objekt (obj) importiert. Ein bereits bestehender Inhalt wird dabei überschrieben. Die externe Datei darf nicht leer oder größer als 2 GB sein.



Der Import von Oberflächenressourcen, wie Dialogen und Menüs ist derzeit nicht möglich.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:

<u>ErrRights</u>	Keine ausreichenden Rechte
<u>ErrFsiNoPath</u>	Pfad im Namen der externen Datei (alpha1) nicht vorhanden
<u>ErrFsiNoFile</u>	Datei im Namen der externen Datei (alpha1) nicht vorhanden
<u>ErrFsiOpenOverflow</u>	Maximale Anzahl offener Dateien erreicht
<u>ErrFsiAccessDenied</u>	Zugriff auf externe Datei (alpha1) verweigert
<u>ErrFsiHdlInvalid</u>	Datei-Deskriptor von externer Datei (alpha1) ungültig
<u>ErrFsiDriveInvalid</u>	Laufwerk im Namen der externen Datei (alpha1) ungültig
<u>ErrFsiSharingViolation</u>	Zugriffskonflikt bei Zugriff auf externe Datei (alpha1)
<u>ErrFsiLockViolation</u>	Sperrkonflikt bei Zugriff auf externe Datei (alpha1)
<u>ErrFsiOpenFailed</u>	Externe Datei (alpha1) konnte nicht geöffnet werden
<u>ErrStoNoData</u>	Die externe Datei enthält keine Daten
<u>ErrStoLocked</u>	Das Storage-Objekt ist gesperrt
<u>ErrStoOperation</u>	Es wird versucht eine Oberflächenressource zu importieren
<u>ErrStoInvalidFormat</u>	Es wurde versucht ein ungültiges Format zu importieren
<u>rDeadlock</u>	Verklemmung aufgetreten

Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt (obj) ungültig

obj -> StoImportTile(alpha1, int2, int3, int4, int5[,  
int6[, int7]]) : int



Kachelgrafik als Storage-Objekt importieren

obj Objekt

alpha1 Pfad und Name einer externen  
Datei

Typ der Kachelgrafik

StoTypeTile Kachelgrafik  
für  
Schaltflächen-  
und  
Listenobjekte

int2 StoTypeTileMenu Kachelgrafik  
für  
Menüobjekte

StoTypeTileTree Kachelgrafik  
für  
Baumobjekte

int3 Transparenzfarbe

int4 Schattenfarbe

int5 Lichtfarbe

int6 Kachelbreite (optional)

int7 Kachelhöhe (optional)

Resultat int Fehlerwert (siehe Text)



Verwandte Befehle,  
StoWriteTileMem(), StoImport(),

Siehe

StoOpen(), Import von  
Kachelgrafiken (Blog)

Mit dieser Funktion wird der Inhalt der externen Kachelgrafik (alpha1) in das Storage-Objekt (obj) importiert. Ein bereits bestehender Inhalt wird dabei überschrieben. Die externe Datei darf nicht leer oder größer als 2 GB sein.

Als Typ der Kachelgrafik (int2) muss eine der folgenden Konstanten angegeben werden:

StoTypeTile Kachelgrafik für Schaltflächen- und Listenobjekte

StoTypeTileMenu Kachelgrafik für Menüobjekte

StoTypeTileTree Kachelgrafik für Baumobjekte

In den Argumenten Transparenzfarbe (int3), Schattenfarbe (int4) und Lichtfarbe (int5) müssen die Farben angegeben werden, die bei der Anzeige der Kachel durch die entsprechenden Systemfarben ersetzt werden. Die Transparenzfarbe wird durch den Hintergrund ersetzt. Die Schattenfarbe und die Lichtfarbe werden durch WinColBtnShadow und WinColBtnHighLight ersetzt.

Bei Bildern mit Alpha-Kanal (32 Bit Farbtiefe) werden diese Farben nicht berücksichtigt.

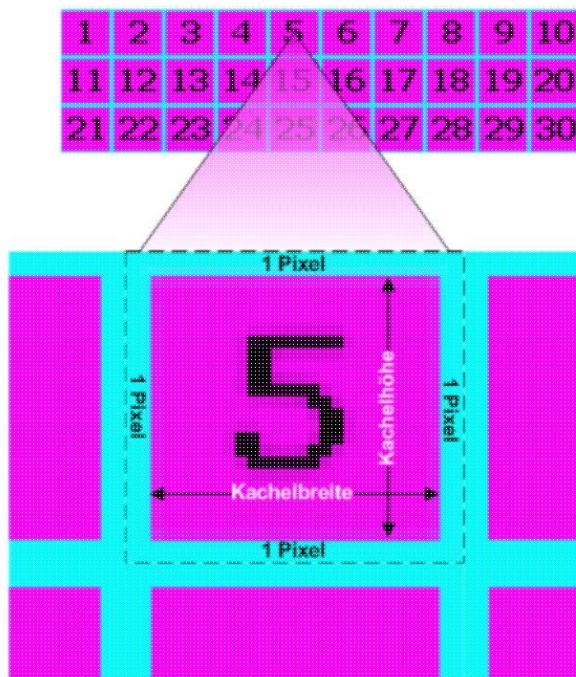
Optional können die Breite einer Kachel (int6) und die Höhe einer Kachel (int7)

## Kontakt

angegeben werden. Werden diese Argumente nicht gesetzt, werden folgende Standardabmessungen verwendet:

Typ	Breite	Höhe
<u>StoTypeTile</u>	24 Pixel	24 Pixel
<u>StoTypeTileMenu</u>	16 Pixel	16 Pixel
<u>StoTypeTileTree</u>	16 Pixel	16 Pixel

Die Größe wird ohne Rand angegeben. Bei Kachelgrafiken für Schaltflächen-, Listen- und Menü-Objekte muss um jede Kachel ein Rand von einem Pixel sein. Somit ist zwischen zwei Kacheln ein Rand von 2 Pixeln. Siehe Bild:



Folgende Fehlerwerte werden von der Funktion zurückgegeben:

<u>ErrRights</u>	Keine ausreichenden Rechte
<u>ErrFsiNoPath</u>	Pfad im Namen der externen Datei (alpha1) nicht vorhanden
<u>ErrFsiNoFile</u>	Datei im Namen der externen Datei (alpha1) nicht vorhanden
<u>ErrFsiOpenOverflow</u>	Maximale Anzahl offener Dateien erreicht
<u>ErrFsiAccessDenied</u>	Zugriff auf externe Datei (alpha1) verweigert
<u>ErrFsiHdlInvalid</u>	Datei-Deskriptor von externer Datei (alpha1) ungültig
<u>ErrFsiDriveInvalid</u>	Laufwerk im Namen der externen Datei (alpha1) ungültig
<u>ErrFsiSharingViolation</u>	Zugriffskonflikt bei Zugriff auf externe Datei (alpha1)
<u>ErrFsiLockViolation</u>	Sperrkonflikt bei Zugriff auf externe Datei (alpha1)
<u>ErrFsiOpenFailed</u>	Externe Datei (alpha1) konnte nicht geöffnet werden
<u>ErrStoNoData</u>	Die externe Datei enthält keine Daten
<u>ErrStoLocked</u>	Das Storage-Objekt ist gesperrt
<u>ErrStoOperation</u>	

## Kontakt

Es wird versucht ein Tile in ein falsches Verzeichnis zu importieren

ErrStoInvalidFormat Es wurde versucht ein ungültiges Format zu importieren  
rDeadlock Verklemmung aufgetreten

Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt (obj) ungültig

ErrValueInvalid Typ der Kachelgrafik ist nicht StoTypeTile, StoTypeTileMenu oder StoTypeTileTree

## Kontakt

obj ->

StoOpen(alpha1[,  
int2]) : handle



Storage-Objekt öffnen

obj

Deskriptor des

Storage-Verzeichnisses

alpha1 Name des Storage-Objekts

Optionen (optional)

\_StoDirectory Verzeichnis

int2 öffnen

\_StoCreate Objekt erstellen


\_StoDbas? Datenbankbereich

Deskriptor des

Resultat handle Storage-Objekts  
oder Fehlerwert

Verwandte Befehle,

Siehe StoClose(), Eigenschaften  
eines Storage-Objekts

Diese Anweisung öffnet ein Storage-Objekt  in dem angegebenen Verzeichnis. Das Verzeichnis muss zuvor mit der Anweisung StoDirOpen() geöffnet wurden sein. Der von diesem Befehl zurückgegebene Deskriptor wird in (obj) übergeben.

Der Name des zu öffnenden Storage-Objekts wird in (alpha1) angegeben. Ist der Name des Objekts nicht bekannt, kann er durch StoDirRead() ermittelt werden.

Ist der Name des Storage-Objekts bekannt, kann auch der vollständige Pfad in (alpha1) angegeben werden (zum Beispiel 'Menu\MnAppMain'). Ein Storage-Verzeichnis muss dann nicht geöffnet werden und wird mit 0 übergeben werden.

Folgende Optionen (int2) können angegeben werden:

\_StoDirectory Der Befehl öffnet ein Verzeichnis (siehe StoDirOpen()).

\_StoCreate Das Objekt wird im Ausgangsverzeichnis erstellt. Storage-Objekte können nicht im Wurzelverzeichnis erstellt werden.

\_StoDbas? Objekt wird in einer mit DbasConnect() verbundenen Datenbank geöffnet oder erstellt. Der Datenbankbereich wird in der Option mit \_StoDbas2 bis \_StoDbas8 angegeben.

Der zurückgegebene Deskriptor kann verwendet werden, um die Eigenschaften des Storage-Objekts zu ermitteln.

Im Fehlerfall wird eine der folgenden Konstanten zurückgegeben:

ErrRights Keine ausreichenden Rechte

ErrStoNameInvalid Der Name des Storage-Objekts ist ungültig.

ErrStoNoFile Der in (alpha1) angegebene Verzeichniseintrag ist nicht vorhanden.

rDeadlock Verklemmung aufgetreten

## Kontakt

obj -> StoReadMem(handle1) : int         Storage-

Objekt in Memory-Objekt lesen

obj Deskriptor eines

Storage-Objektes

handle1 Deskriptor eines

Memory-Objektes

Resultat int Fehlerwert 

Verwandte  
Befehle,

Siehe

StoExport(),  
StoOpen()

Mit dieser Funktion wird der Inhalt des Storage-Objektes (obj) in das Memory-Objekt (handle1) eingelesen. Das Resultat ist ErrOk.

Der Wert der Eigenschaft Len entspricht nach der Operation der unkomprimierten Datengröße des Storage-Objektes.



Das Exportieren von Resource-Elementen wie Dialogen und Menüs ist derzeit nicht möglich.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:

ErrRights

Keine ausreichenden Rechte

ErrStoNoData

Das Storage-Objekt enthält keine Daten Mögliche

Laufzeitfehler:

ErrHdlInvalid

Der in (obj) oder (handle1) übergebene Deskriptor ist ungültig.

ErrStringOverflow


Das zu lesenden Storage-Objekt ist größer als das  
Memory-Objekt.

## Kontakt

obj -> StoWriteMem(handle1) : int        Storage-Objekt  
aus Memory-Objekt schreiben

obj Deskriptor eines  
Storage-Objektes

handle1 Deskriptor eines  
Memory-Objektes

Resultat int Fehlerwert (siehe Text)  —

Verwandte Befehle,

Siehe StoImport(), StoReadMem(),  
StoOpen()

Mit dieser Funktion wird der komplette Inhalt des Memory-Objekts (handle1) in das Storage-Objekt (obj) geschrieben. Ein bereits bestehender Inhalt wird dabei überschrieben.



Der Import von Oberflächenressourcen, wie Dialogen und Menüs ist derzeit nicht möglich.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:

<u>ErrRights</u>	Keine ausreichenden Rechte
<u>ErrStoNoData</u>	Das Memory-Objekt enthält keine Daten
<u>ErrStoLocked</u>	Das Storage-Objekt ist gesperrt
<u>ErrStoOperation</u>	Es wird versucht eine Oberflächenressource zu importieren
<u>ErrStoInvalidFormat</u>	Es wurde versucht ein ungültiges Format zu importieren
<u>rDeadlock</u>	Verklemmung aufgetreten

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder (handle1) angegebene Deskriptor ist ungültig.



## Kontakt

obj -> StoWriteTileMem(handle1, int2, int3, int4, int5[, int6[, int7]]) : int



**Kachelgrafik als Storage-Objekt aus Memory-Objekt schreiben**

obj Deskriptor eines

Storage-Objektes

handle1 Deskriptor eines

Memory-Objektes

Typ der Kachelgrafik

\_StoTypeTile

Kachelgrafik  
für  
Schaltflächen-  
und  
Listenobjekte

int2

\_StoTypeTileMenu Kachelgrafik

für  
Menüobjekte

\_StoTypeTileTree Kachelgrafik

für  
Baumobjekte

int3 Transparenzfarbe

int4 Schattenfarbe

int5 Lichtfarbe

int6 Kachelbreite (optional)

int7 Kachelhöhe (optional)

Resultat int Fehlerwert (siehe Text)



Verwandte Befehle,

Siehe StoImportTile(), StoWriteMem(),

StoOpen()

Mit dieser Funktion wird der komplette Inhalt des Memory-Objekts (handle1) in das Storage-Objekt (obj) geschrieben. Ein bereits bestehender Inhalt wird dabei überschrieben.

Als Typ der Kachelgrafik (int2) muss eine der folgenden Konstanten angegeben werden:

\_StoTypeTile Kachelgrafik für Schaltflächen- und Listenobjekte

\_StoTypeTileMenu Kachelgrafik für Menüobjekte

\_StoTypeTileTree Kachelgrafik für Baumobjekte

In den Argumenten Transparenzfarbe (int3), Schattenfarbe (int4) und Lichtfarbe (int5) müssen die Farben angegeben werden, die bei der Anzeige der Kachel durch die entsprechenden Systemfarben ersetzt werden. Die Transparenzfarbe wird durch den Hintergrund ersetzt. Die Schattenfarbe und die Lichtfarbe werden durch \_WinColBtnShadow und \_WinColBtnHighLight ersetzt.

Bei Bildern mit Alpha-Kanal (32 Bit Farbtiefe) werden diese Farben nicht berücksichtigt.

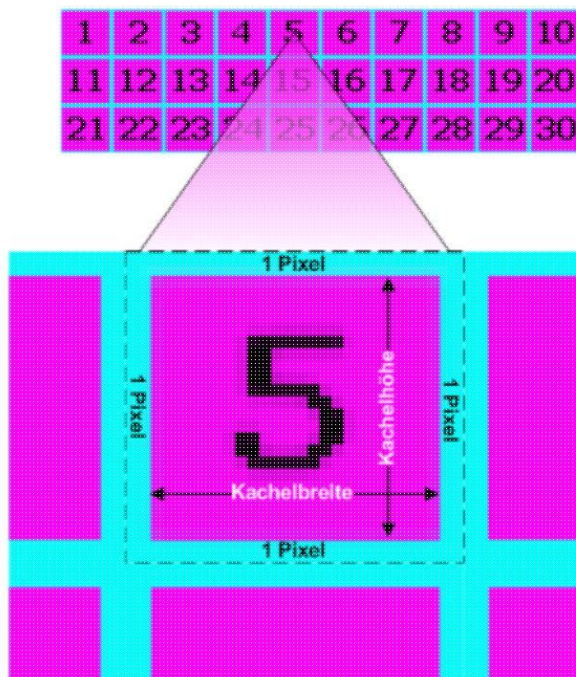
Optional können die Breite einer Kachel (int6) und die Höhe einer Kachel (int7)

## Kontakt

angegeben werden. Werden diese Argumente nicht gesetzt, werden folgende Standardabmessungen verwendet:

Typ	Breite	Höhe
<u>_StoTypeTile</u>	24 Pixel	24 Pixel
<u>_StoTypeTileMenu</u>	16 Pixel	16 Pixel
<u>_StoTypeTileTree</u>	16 Pixel	16 Pixel

Die Größe wird ohne Rand angegeben. Bei Kachelgrafiken für Schaltflächen-, Listen- und Menü-Objekte muss um jede Kachel ein Rand von einem Pixel sein. Somit ist zwischen zwei Kacheln ein Rand von 2 Pixeln. Siehe Bild:



Folgende Fehlerwerte werden von der Funktion zurückgegeben:

<u>_ErrRights</u>	Keine ausreichenden Rechte
<u>_ErrStoNoData</u>	Das <u>Memory-Objekt</u> enthält keine Daten
<u>_ErrStoLocked</u>	Das <u>Storage-Objekt</u> ist gesperrt
<u>_ErrStoOperation</u>	Es wird versucht ein Tile in ein falsches Verzeichnis zu importieren
<u>_ErrStoInvalidFormat</u>	Es wurde versucht ein ungültiges Format zu importieren
<u>_rDeadlock</u>	Verklebung aufgetreten

Mögliche Laufzeitfehler:

<u>_ErrHdlInvalid</u>	Der in (obj) oder (handle1) angegebene Deskriptor ist ungültig.
<u>_ErrValueInvalid</u>	Typ der Kachelgrafik ist nicht <u>_StoTypeTile</u> , <u>_StoTypeTileMenu</u> oder <u>_StoTypeTileTree</u>

Konstanten für Storage-Objekte

Konstanten für Storage-Objekte

Siehe Befehle für  
Storage-Objekte

- StoCreate
- StoDb2
- StoDb3
- StoDb4
- StoDb5
- StoDb6
- StoDb7
- StoDb8
- StoDirectory
- StoFirst
- StoLast
- StoNext
- StoPrev

## Kontakt

**\_StoCreate**

**Storage-Objekt erstellen**

Wert 4.096 /

**0x00001000**

**Verwandte**

Siehe **Befehle**,

**StoOpen()**

Option bei **StoOpen()** durch die ein neues Storage-Objekt erstellt werden kann.

## Kontakt

**\_StoDb?**

Zugriff auf Storage-Objekte in verbundenen Datenbanken Wert

**0x10000**  
**0x70000** -

Siehe **StoDirOpen()**  
**StoOpen()**,

Option bei **StoDirOpen()** und **StoOpen()** durch die ein Storage-Objekt bzw.  
-Verzeichniss in einer anderen Datenbank angesprochen werden kann.

Zuvor muss diese Datenbank mit dem Befehl **DbConnect()** verbunden werden. Der dabei  
angegebene Nummernbereich bestimmt, mit welcher Option Storage-Objekte/-Verzeichnisse dieser  
Datenbank angesprochen werden können:

Nummernbereich 2 : **\_StoDb2**

Nummernbereich 3 : **\_StoDb3**

Nummernbereich 4 : **\_StoDb4**

Nummernbereich 5 : **\_StoDb5**

Nummernbereich 6 : **\_StoDb6**

Nummernbereich 7 : **\_StoDb7**

Nummernbereich 8 : **\_StoDb8**

## Kontakt

**\_StoDirectory**

**Storage-Verzeichnis öffnen**

Wert 8.192 /

**0x00002000**

Siehe StoOpen()

Bei der Anweisung StoOpen() kann bei den Optionen **\_StoDirectory** angegeben werden. Der Befehl entspricht dann der Anweisung StoDirOpen().

## Kontakt

**\_StoFirst**

**Erstes Storage-Objekt lesen**

**Wert** <sup>1/</sup>

**0x00000001**

**Siehe StoDirRead()**

**Mit dieser Option bei der Anweisung StoDirRead() wird der erste Verzeichniseintrag gelesen.**

## Kontakt

**\_StoPrev**

**Vorhergehendes Verzeichnis lesen**

**Wert**<sup>3/</sup>

**0x00000003**

**Siehe StoDirRead()**

**Mit dieser Option bei der Anweisung StoDirRead() wird der vorherige Verzeichniseintrag gelesen.**



## Kontakt

**\_StoNext**

**Nächstes Verzeichnis lesen**

Wert<sup>4/</sup>

**0x00000004**

Siehe **StoDirRead()**

Mit dieser Option bei der Anweisung **StoDirRead()** wird der nächste Verzeichniseintrag gelesen.

## Kontakt

**\_StoLast**

**Letztes Verzeichnis lesen**

Wert<sup>2/</sup>

**0x00000002**

**Siehe StoDirRead()**

**Mit dieser Option bei der Anweisung StoDirRead() wird der letzte Verzeichniseintrag gelesen.**

### Selektionen

Teilmengen von Datensätzen und deren Verarbeitung

Befehlsgruppen,

Befehlsliste,

Befehle für

Siehe dynamische

Selektionen,

Möglichkeiten der

Datensatzfilterung

(Blog)

Eine Selektionsmenge ist eine Teilmenge von Datensätzen aus einer oder mehreren Dateien. In einer Selektion werden Datensätze auf bestimmte Bedingungen überprüft und gültige Sätze in das Selektionsresultat aufgenommen. Das Resultat einer Selektion enthält Verweise auf bestehende Datensätze, d. h. die Sätze in einer Selektion sind nicht redundant gespeichert. Eine Selektion kann mehrere Ergebnismengen beinhalten, die Verweise auf Sätze verschiedener Dateien enthalten. Die Hauptergebnismenge (Datei, in der die Selektion definiert ist) lässt sich beliebig sortieren. Weitere Ergebnismengen enthalten Verweise auf Daten verknüpfter Dateien und sind nach den entsprechenden Verknüpfungsschlüsseln sortiert.

Zunächst können in einer Selektion die Felder der Datei auf verschiedenste Kriterien hin abgefragt werden. Desweiteren existieren Abfragen auf verknüpfte Datensätze und deren Felder. Möglich sind ebenfalls Berechnungen in der Abfrage, sowie die Einbindung von Prozeduren. Nach der Definition einer Selektion wird diese in Zwischencode (P-Code) übersetzt und kann anschließend durchgeführt werden, wobei die jeweilige Selektionsmenge ermittelt wird. Bei der Durchführung werden automatisch Optimierungen (z. B. die Benutzung von Schlüsseln) benutzt, um die Verarbeitungszeit zu reduzieren (siehe auch Selektionen optimieren).

Auf die Selektionsmenge lässt sich über bestimmte Oberflächen-Objekte und mit Datensatz-Operationen zugreifen. Die Menge kann auch als Grundlage für weitere Selektionen dienen und mit anderen Mengen kombiniert werden.

Eine Selektion kann immer dann verwendet werden, wenn bestimmte Operationen nur auf eine Teilmenge der Datensätze erfolgen sollen.

Dieses Kapitel gliedert sich in folgende Abschnitte:

- Verwendung von Selektionen
- Definition von Selektionen
- Durchführen von Selektionen
- Selektionsmenge verarbeiten
- Selektion löschen
- Verarbeitungshinweise
- Befehle
- Konstanten

### Verwendung von Selektionen

Neben den Selektionen kann noch mit Hilfe von Verknüpfungen oder Filtern die Menge der Datensätze eingeschränkt werden. Selektionen sollten verwendet werden,

## Kontakt

wenn die Treffermenge gering ausfällt und keine Verknüpfung verwendet werden kann.

### Bedingungen

Alle einschränkenden Felder befinden sich in einem Schlüssel.

-> Objekt

-> Verknüpfung

Es wird nur auf einen Schlüsselwert eingeschränkt.

Alle einschränkenden Felder befinden sich in einem Schlüssel.

-> Filter

Es gibt mehrere gültige Schlüsselwerte (zum Beispiel einen Bereich).

Die Bedingungen müssen nicht geklammert werden.

-

-> Selektion

Umfasst das Selektionskriterium genau einen Wert (zum Beispiel: "Suche alle Ansprechpartner deren Kundennummer gleich 1000 ist") und das entsprechende Feld wird in einem Schlüssel verwendet, sollte eine Verknüpfung verwendet werden. Die Datensätze der Verknüpfung stehen sofort zur Verfügung und es muss keine Selektion angelegt oder durchgeführt werden.

Wird nicht nur ein Wert, sondern ein Wertebereich gesucht, kann die Verknüpfung nicht verwendet werden. Befindet sich der Wert in einem Schlüsselfeld, kann statt dessen über einen Filter zugegriffen werden. Befindet sich in der Datenmenge nur ein geringer Anteil an Datensätzen, die das Kriterium erfüllen, kann der Zugriff über einen Filter langsam erscheinen, da alle Datensätze zwischen den Treffern ebenfalls gelesen werden.

Besteht das Selektionskriterium aus mehreren Bedingungen, die einen Wertebereich abfragen und mit unterschiedlichen logischen Operatoren verknüpft sind (UND, ODER, NICHT, ...), muss eine Selektion verwendet werden. Die Verknüpfung kann keinen Wertebereich abfragen und der Filter kann Ausdrücke nicht klammern.

### Definition von Selektionen

Eine Selektion kann mit Hilfe von Befehlen zur Laufzeit des Programms erstellt werden. Eine Liste der Befehle befindet sich im Abschnitt Befehle für dynamische Selektionen. Die Definition der Selektion erfolgt zunächst in einem Selection-Objekt. Dieses Objekt wird mit der Anweisung SelCreate() angelegt.

Im einfachsten Fall werden beim Anlegen des Objekts eine Datei und ein Schlüssel zur Sortierung der Datensätze angegeben. Anschließend kann mit der Anweisung SelDefQuery() die Bedingung angegeben werden. Die Bedingung ist ein logischer Ausdruck, d. h. ein Ausdruck mit einem Ergebnis von true oder false. Zur Definition dieser Ausdrücke siehe Logische Ausdrücke in dynamischen Selektionen.

Die Sortierung der Datensätze muss nicht über einen Schlüssel erfolgen. Wird beim Anlegen des Objekts kein Schlüssel angegeben, kann später bei der Durchführung der Selektion ein beliebiges Feld zur Sortierung verwendet werden (siehe auch SelKeyMode).

Ist die Definition abgeschlossen, wird die Selektion in der Datenbank gespeichert (siehe SelStore()). Ab diesem Zeitpunkt steht die Selektion zur Durchführung zur Verfügung. Die Selektionsmenge (die Datensätze, die die Bedingung erfüllen) ist zu

## Kontakt

diesem Zeitpunkt noch leer. Die Überprüfung der Datensätze, ob sie die Bedingung erfüllen, findet erst bei der Durchführung der Selektion statt.

Beispiel:

In der folgenden Prozedur wird ein Selection-Objekt angelegt und als Selektion in der Datenbank gespeichert. Die Selektion wird mit dem angegebenen Namen in der Datenbank gespeichert. Der Name darf höchstens 20 Zeichen lang sein. Die Selektionsmenge enthält keine Datensätze, da die Selektion noch nicht durchgeführt wurde.

```
main local {      tSel : handle;      tErr : int;  }{ tSel # SelCreate(tblArtArticle, keyArtNumber
```

Bei der Definition und der Speicherung einer Selektion muss der zurückgegebene Fehlerwert ausgewertet werden. In dem Beispiel wurde das aus Gründen der Übersichtlichkeit weggelassen.

Die Zeichenkette des Selektionskriteriums ('ffArtPrice > 10.0') wurde in diesem Fall statisch angegeben. Sie kann auch aus Angaben des Benutzers zusammengesetzt werden:

```
... tErr # tSel->SelDefQuery('', tFieldName + ' > ' + CnvAF(tPriceValue)); ...
```

Soll die Selektion sofort weiter verwendet werden, kann sie beim Speichern gleich gesperrt werden. Das Selection-Objekt muss ebenfalls nicht geschlossen werden. Statt dessen wird das Objekt in einen Selektionsdeskriptor umgewandelt:

```
... tErr # tSel->SelStore('TMP.Articleprice', _SelLock); tSel # tSel->SelOpen(); ...
```

Der Selektionsdeskriptor muss bei den weiteren Anweisungen angegeben werden.

Innerhalb einer Datei muss der Name der Selektion eindeutig sein. Der in dem Beispiel angegebene Name ist statisch definiert, d. h. führen zwei Clients die gleiche Prozedur aus, kommt es zu einem Fehler, da der zweite Benutzer die Selektion nicht unter dem gleichen Namen speichern kann. Sollen mehrere Benutzer die Prozedur ausführen können, kann der Name der Selektion mit der Benutzer-Id erweitert werden.

```
... tErr # tSel->SelStore(UserInfo(_UserCurrent) + '.Articleprice', _SelLock); ...
```

Die verwendete Benutzer-Id ist für jeden angemeldeten Benutzer eindeutig. Soll innerhalb eines gestarteten Clients die Prozedur mehrfach gestartet werden, muss ein zusätzlicher Zähler verwendet werden, damit die Namen eindeutig sind.

### Durchführen von Selektionen

Damit eine Selektion durchgeführt werden kann, muss zunächst ein Selektionspuffer mit der Anweisung SelOpen() angelegt werden. Anschließend kann die entsprechende Selektion gelesen werden (siehe SelRead()). Ab diesem Zeitpunkt steht die Menge der Datensätze, die beim letzten Selektionsdurchlauf gefunden wurden, zur Verfügung. Werden aktuelle Daten benötigt, muss die Selektion gestartet werden. Bei allen

## Kontakt

Operationen, die die Selektion oder die Selektionsmenge ändern können, muss die Selektion zuvor mit einer Sperroption ( SelLock ) geöffnet werden.

Steht ein Selektionsdeskriptor mit einer gesperrten Selektion zur Verfügung, kann die Selektion mit der Anweisung SelRun() durchgeführt werden. Bei der Durchführung werden Datensätze aus der Datei gelesen und mit dem Selektionskriterium verglichen. Erfüllt der Datensatz das Kriterium, wird ein Verweis auf den Datensatz in der Selektionsmenge gespeichert.

```
... tErr # tSel->SelRun(_SelDisplay | _SelWait); ...
```

Die Durchführung einer Selektion kann abhängig von der Menge der zu durchsuchenden Datensätze und vom Aufbau des Selektionskriteriums einige Zeit in Anspruch nehmen. Hinweise zur Verbesserung des Laufzeitverhaltens von Selektionen befinden sich im Abschnitt Selektionen optimieren.

In diesem Beispiel wird der Fortschritt der Selektion angezeigt und nach der Durchführung auf eine Eingabe des Benutzers gewartet.

Weitere Parameter sind bei der Anweisung SelRun() beschrieben. Nach der Durchführung der Selektion kann auf die Selektionsmenge zugegriffen werden, bis der Selektionsdeskriptor geschlossen wird.

### Selektionsmenge verarbeiten

Um auf die Datensätze einer Selektionsmenge zugreifen zu können, wird ein Selektionsdeskriptor benötigt (SelOpen()) und die entsprechende Selektionsmenge muss gelesen worden sein (SelRead()). Sollen an der Selektionsmenge Änderungen vorgenommen werden (Datensätze löschen oder hinzufügen), muss die Selektionsmenge sperrend gelesen werden.



Änderungen an den Datensätzen können ohne Sperrung der Selektionsmenge erfolgen, da in der Selektionsmenge nur Referenzen auf die Datensätze enthalten sind. Werden Datensätze nach der Durchführung der Selektion verändert, wirkt sich das nicht auf die Zugehörigkeit zur Selektionsmenge aus.

Der Inhalt der Selektionsmenge kann in einem RecList-, RecListPopup- oder PrintDocRecord-Objekt ausgegeben werden. Der Selektionsdeskriptor muss dazu der Eigenschaft DbSelection zugewiesen werden.

```
... $RecList->wpDbSelection # tSel; ...
```

Dabei ist zu beachten, dass in der Eigenschaft DbFileNo die gleiche Dateinummer angegeben sein muss, in der die Selektion definiert wurde. Der Selektionsdeskriptor darf erst geschlossen werden, wenn die Objekte nicht mehr auf die Selektionsmenge zugreifen müssen. Sollen die Datensätze prozedural gelesen werden, kann bei der Anweisung RecRead() der Selektionsdeskriptor anstelle des Schlüssels angegeben werden.

```
... for tErr # RecRead(tblArtArticle, tSel, _RecFirst) loop tErr # RecRead(tblArtArticle,
```

## Kontakt

In diesem Beispiel werden alle Datensätze in der Selektion gelesen.

Einzelne Datensätze können mit den Anweisungen SelRecDelete() und SelRecInsert() aus einer Selektionsmenge gelöscht oder eingefügt werden. Mit diesen Anweisungen können komplette Selektionsmengen ohne ein Selektionskriterium aufgebaut werden. Die Anweisungen können nur verwendet werden, wenn die Selektionsmenge sperrend gelesen wurde.

Wird nach der Verarbeitung die Selektionsmenge nicht mehr benötigt, kann der Selektionsdeskriptor mit der Anweisung SelClose() geschlossen werden.

### Selektion löschen

Wird eine Selektion nicht mehr benötigt, kann sie mit der Anweisung SelDelete() aus der Datenbank entfernt werden.

```
... tErr # SelDelete(tblArtArticle, UserInfo(_UserCurrent) + '.Articleprice'); ...
```

### Verarbeitungshinweise

Eine Selektionsmenge ist eine Momentaufnahme des Datenbestandes. Werden nach dem Durchführen einer Selektion Änderungen am Datenbestand vorgenommen, werden diese nicht berücksichtigt, bis die Selektion erneut durchgeführt wird. Der Zeitpunkt, zudem eine Selektion zuletzt durchgeführt wurde, kann mit den Anweisungen SelInfoDate() und SelInfoTime() ermittelt werden.

Werden Datensätze, die in der Selektionsmenge enthalten sind, gelöscht, wirkt sich das nicht auf die Verarbeitung der Selektionsmenge aus. Befindet sich in einer Selektionsmenge eine Referenz auf einen nicht vorhandenen Datensatz, wird dieser übersprungen. Neu angelegte Datensätze werden keiner Selektionsmenge zugeordnet.

Selektionsmengen werden in der Datenbank gespeichert. Dass heißt nach dem Durchführen der Selektion kann auch ohne eine erneute Durchführung auf die zuletzt selektierten Datensätze zugegriffen werden. Diese Selektionsmengen können mit der Anweisung SelClear() geleert werden. Wird eine Selektion gelöscht (siehe SelDelete()), steht auch die Selektionsmenge nicht mehr zur Verfügung. Durch die Durchführung einer Optimierung werden alle Selektionsmengen geleert.

### Befehle

- SelClear
- SelClose
- SelCopy
- SelDelete
- SelIgnore
- SelInfo
- SelInfoAlpha
- SelInfoDate
- SelInfoTime
- SelOpen
- SelRead

- SelRecDelete
- SelRecInsert
- SelRun
- SelValue

## Konstanten

- SelAvg
- SelAvgD
- SelBase
- SelBreak
- SelCount
- SelCountD
- SelCreated
- SelDisplay
- SelExecuted
- SelFile
- SelFirst
- SelInter
- SelKeyMode
- SelKeyUpdate
- SelLast
- SelLock
- SelMax
- SelMin
- SelMinus
- SelModified
- SelName
- SelNext
- SelPrev
- SelRemarks
- SelResultSet
- SelServer
- SelServerAllFld
- SelServerAutoFld
- SelSharedLock
- SelSort
- SelSum
- SelSumD
- SelUnion
- SelUnlock
- SelUser
- SelWait



## Kontakt

### Selektionen optimieren

#### Hinweise zum Aufbau effizienter Selektionen

Die Durchführung einer Selektion (siehe SelRun()) kann abhängig vom Selektionskriterium und von der Anzahl zu durchsuchender Datensätze eine längere Laufzeit in Anspruch nehmen. Um Selektionen zu beschleunigen gibt es verschiedene Möglichkeiten, die in diesem Abschnitt erläutert werden.

Genereller Ablauf bei der Durchführung einer Selektion:

1. Client liest den ersten Datensatz aus der Datenbank
2. Überprüfung des Selektionskriteriums
3. Server mitteilen, ob der Datensatz in die Selektionsmenge gehört
4. Lesen des nächsten Datensatzes und weiter bei 2

Es gibt unterschiedliche Möglichkeiten eine Selektion zu optimieren:

- Selektionsmenge ohne Sortierung
- Verwendung von Schlüsseln (Vorauswahlen)
- Durchführung der Selektion beim Server
- Umstellung der Selektionskriterien
- Verwendung einer anderen Ausgangsdatei
- Verwendung ohne SelRun()

#### Selektionsmenge ohne Sortierung

Kann auf die Sortierung der Selektionsmenge verzichtet werden, ist das Einfügen von Datensätzen in die Selektionsmenge schneller. Das wirkt sich besonders dann aus, wenn eine große Anzahl von Datensätzen zur Treffermenge gehören.

Wird keine Sortierung benötigt, wird beim Erstellen der Selektion (siehe SelCreate()) kein Schlüssel angegeben.

#### Verwendung von Schlüsseln (Vorauswahlen)

Der CONZEPT 16-Server kann bei der Durchführung bereits Optimierungen vornehmen. Befindet sich der zu überprüfende Wert in einem Schlüssel, kann eine Vorauswahl der zu überprüfenden Datensätze erfolgen. Es müssen somit nicht mehr alle Datensätze vom Client gelesen werden.

#### Beispiel:

Es sollen alle Artikel ermittelt werden, die einen bestimmten Preis übersteigen. Es wird eine entsprechende Selektion angelegt.

#### Selektion ohne Schlüssel

#### Selektion mit Schlüssel

Durch das Anlegen eines Schlüssels müssen weniger Datensätze gelesen werden. Dadurch beschleunigt sich die Selektion erheblich.

## Kontakt



Bei der Verwendung von Vergleichen ohne die Berücksichtigung der Groß-/Kleinschreibung kann nur dann durch den Server eine Vorauswahl getroffen werden, wenn ein entsprechender Schlüssel mit Groß-/Kleinwandlung vorhanden ist. Bei Vergleichen auf Ähnlichkeit (=\* und =\*^ ) kann nur dann eine Vorauswahl getroffen werden, wenn die Platzhalter am Ende der Zeichenkette stehen.

Bei der Definition der Selektion kann der zu verwendende Schlüssel für die Vorauswahl bei der Anweisung SelDefQuery() im Selektionskriterium mit {...} angegeben werden (siehe Logische Ausdrücke in dynamischen Selektionen).

Unter bestimmten Bedingungen reicht der Zugriff auf den Schlüssel aus und der Datensatz muss nicht gelesen werden. Das ist dann der Fall, wenn die Selektionsmenge nicht sortiert ist (siehe oben), für alle Selektionskriterien eine Vorauswahl verwendet werden kann und der Inhalt des Datensatzes nicht für weitere Selektionskriterien benötigt wird (zum Beispiel zum Lesen von verknüpften Datensätzen). Unter diesen Umständen kann angegeben werden, dass nur die Vorauswahl verwendet werden soll. Im Selektionskriterium wird das durch {...+} angegeben. Die Zeitersparnis wirkt sich erst bei einer größeren Anzahl von Datensätze in der Treffermenge aus.

### Durchführung der Selektion beim Server

Alle zu überprüfenden Datensätze werden von der Datenbank zum Client übertragen. Diese Übertragung ist nicht notwendig, wenn die Selektion direkt beim Server ausgeführt wird. Das kann bei der Anweisung SelRun() angegeben werden. Die Selektion wird dort mit einem neuen Benutzer durchgeführt. Folgende Parameter stehen zur Verfügung:

#### SelServer

Ausführung auf dem Server

#### SelServerAutoFld

Ausführung auf dem Server, Übertragen der Feldinhalte der verwendeten Felder

#### SelServerAllFld

Ausführung auf dem Server, Übertragen aller nicht leeren Feldinhalte der Datenstruktur

Die Optionen unterscheiden sich darin, welche Feldpuffer vor der Durchführung der Selektion an den Server übertragen werden. Greift die Selektion nicht auf Feldpuffer zu (d. h. in den Kriterien werden keine zwei Felder miteinander verglichen), kann die Option SelServer verwendet werden. Erfolgt ein Vergleich auf Felder (zum Beispiel `ffArtPrice >= ffSelPriceMin`) muss dieses Feld mit der Option SelServerAutoFld an den Server übertragen werden. Wird in der "Prozedur nach Abfrage" auf Feldinhalte verwiesen, die nicht in den Abfragekriterien enthalten sind, muss die Option SelServerAllFld angegeben werden. Die Übertragung der Feldinhalte benötigt mehr Zeit, je größer die Datenstruktur ist.



Ist eine "Prozedur nach Abfrage" bei der Selektion angegeben, ist darauf zu achten, dass alle Anweisungen innerhalb der Prozedur auch vom Server ausgeführt werden können.

### Umstellung der Selektionskriterien

Bei dieser Form der Optimierung werden Kenntnisse über den Datenbestand benötigt. Werden zwei Kriterien mit "UND" verknüpft, muss das zweite Kriterium nicht mehr

## **Kontakt**

ausgewertet werden, wenn das erste Kriterium bereits feststellt, dass der Datensatz nicht in die Selektionsmenge gehört. Die Anzahl der Vergleiche kann somit reduziert werden, indem das stärker einschränkende Kriterium zuerst überprüft wird.

Erfolgt die Verknüpfung mit "ODER" muss das zweite Kriterium nicht geprüft werden, wenn das erste bereits die Zugehörigkeit zur Selektionsmenge feststellt. In diesem Fall sollte das schwächer einschränkende Kriterium zuerst überprüft werden.

In einer Artikel-Datei sind 100 Sätze gespeichert. Je 50 der Sätze gehören zu einer Artikelgruppe. In der Datei befinden sich nur zwei Artikel mit einem Preis über 100 EUR. In die Selektionsmenge sollen alle Artikel einer Artikelgruppe mit einem Preis größer 100 EUR. Wird zuerst die Artikelgruppe und anschließend der Preis überprüft, werden 150 Vergleiche benötigt (100 Vergleiche für die Artikelgruppe und 50 Vergleiche mit dem Preis), wird zuerst der Preis überprüft werden nur 102 Vergleiche benötigt (100 Vergleiche für den Preis und 2 Vergleiche mit der Artikelgruppe).

Gibt es einen Schlüssel über die Artikelgruppe, kommt man zu einem anderen Ergebnis, da nur noch die Hälfte der Datensätze gelesen wird (50 Vergleiche für die Artikelgruppe und 50 Vergleiche mit dem Preis).

Abfragen, die Vergleiche auf verknüpfte Datensätze beinhalten, sollten zuletzt durchgeführt werden. Wird der Datensatz aufgrund der früheren Kriterien nicht in die Selektionsmenge aufgenommen, müssen die verknüpften Sätze erst gar nicht gelesen werden.

### **Verwendung einer anderen Ausgangsdatei**

Prinzipiell werden bei der Durchführung einer Selektion nacheinander die Sätze der Ausgangsdatei gelesen und das erste Abfragekriterium ausgewertet. Sind weitere Abfragekriterien enthalten, die auf verknüpfte Sätze zugreifen, werden nacheinander die verknüpften Sätze gelesen und für diese das Abfragekriterium ausgewertet.

### **Beispiel:**

Es sollen alle Kunden ermittelt werden, die in einem bestimmten Monat mindestens einen Auftrag hatten. Es wird also eine Selektion in der Kunden-Datei angelegt. Als Abfragekriterium wird "Mindestens ein Datensatz in Abfrage Aufträge" angegeben. Die Abfrage "Aufträge" untersucht die verknüpften Datensätze, ob sie in dem angegebenen Monat liegen. Sind in der Datenbank viele Kunden, aber es wurden in dem betreffenden Monat nur wenige Aufträge ausgeführt, müssen viele Datensätze gelesen werden, um zu dem Ergebnis zu kommen. Zunächst müssen alle Kundendatensätze gelesen werden, da anschließend die Abfrage "Aufträge" überprüft werden kann. Diese Abfrage kann vom Server optimiert werden, indem er nur die Aufträge des betreffenden Monats liest. Sind keine Aufträge vorhanden, ist der Kundensatz unnötigerweise gelesen worden.

Wird die Selektion von der Seite der Aufträge betrachtet, kann der Server zunächst alle Aufträge des betreffenden Monats und anschließend die dazugehörigen Kundendatensätze lesen. Gibt es vergleichsweise wenige Aufträge, müssen dazu wesentlich weniger Datensätze gelesen werden.

## Kontakt

### Beispiel in Zahlen

Anzahl der Kundendatensätze	10.000
Anzahl der Auftragsdatensätze	100.000
Anzahl der Aufträge in diesem Monat	500

Ist die Ausgangsdatei die Kundendatei, müssen alle Kunden gelesen werden (10.000) und davon ermittelt werden, ob ein Auftrag aus dem Monat existiert. Da dieser Zugriff durch den Server optimiert werden kann, sind das nur noch 500 Zugriffe. Zusammen also 10.500 Zugriffe in die Datenbank.

Werden die Aufträge als Ausgangsdatei verwendet, erfolgt zunächst der optimierte Zugriff auf die Aufträge des Monats (500 Zugriffe), und anschließend werden die dazugehörenden Kunden ermittelt (ebenfalls 500 Zugriffe). Zusammen also nur 1.000 Zugriffe in die Datenbank.

Ähnlich wie bei der Umstellung der Selektionskriterien, sollte das am weitesten einschränkende Kriterium zuerst geprüft werden. Liegt dieses Kriterium in einer anderen Datei, sollte diese als Ausgangsdatei verwendet werden.

### Verwendung ohne SelRun()

Selektionsmengen müssen nicht zwingend mit der Anweisung SelRun() mit Datensätzen gefüllt werden. Ist eine Selektion gespeichert, kann die Selektionsmenge mit der Anweisung SelRecInsert() mit Datensätzen gefüllt werden. Die Überprüfung des Selektionskriteriums findet nicht mehr in der Selektion selbst, sondern in der Funktion, in der der Datensatz gespeichert bzw. geändert wird, statt. Sollte der Datensatz gelöscht oder durch eine Änderung nicht mehr das Selektionskriterium erfüllen, kann er mit der Anweisung SelRecDelete() wieder aus der Selektionsmenge entfernt werden.

Dabei ist zu beachten, dass die Selektionsmenge geleert wird, wenn eine Optimierung der Datenbank durchgeführt wird. In diesem Fall muss die Selektionsmenge neu gefüllt werden.

## Kontakt

### Befehle für dynamische Selektionen

Liste der Befehle und Konstanten zur Bearbeitung von Selektionen

Befehlsgruppen,

Siehe Befehlsliste,

Selektionen

#### Befehle

- SelAddLink
- SelAddResult
- SelAddSortFld
- SelCreate
- SelDefQuery
- SelStore

## Kontakt

### Logische Ausdrücke in dynamischen Selektionen

Aufbau und Verwendung von logischen Ausdrücken in dynamischen Selektionen Bei dynamischen Selektionen wird über einen logischen Ausdruck bestimmt, ob ein Datensatz in die Selektionsmenge aufgenommen wird, oder nicht. Dieser Ausdruck hat immer ein logisches Ergebnis (true oder false). Der Ausdruck wird bei der Anweisung SelDefQuery() angegeben.

### Aufbau logischer Ausdrücke

Logische Ausdrücke sind entweder Felder, Variablen oder Konstanten mit einem logischen Wert oder Vergleiche von Feldern, Variablen oder Konstanten beliebigen Typs. Im Zusammenhang mit Selektionen bieten sich hier besonders die Vergleiche von Feldern mit Variablen oder Konstanten an. Da durch das Lesen der Datensätze der Inhalt der Felder mit jedem Satz wechselt, kann auch für jeden Satz das Ergebnis des logischen Ausdrucks unterschiedlich ausfallen.

Beispiele einfacher Ausdrücke mit einem Feld vom Typ logic:

- 'flCstIsVendor'
- '!flCstIsVendor'
- 'flCstIsVendor = true'

Ausdrücke, die keinen logischen Wert besitzen, müssen mit einem anderen Wert gleichen Typs verglichen werden. Dazu stehen eine Reihe von Vergleichsoperatoren zur Verfügung:

'='	gleich
'=^'	gleich (ohne Groß-/Kleinschreibung)
'!='	ungleich
'<'	kleiner
'<='	kleiner oder gleich
'>'	größer
'>='	größer oder gleich
'=*'	Ähnlichkeitsvergleich
'=*^'	Ähnlichkeitsvergleich (ohne Groß-/Kleinschreibung)

'between[]' Wert befindet sich in einem Bereich

Der Ähnlichkeitsvergleich steht nur beim Vergleich vom Typ alpha zur Verfügung. Sollen innerhalb des logischen Ausdrucks eine Zeichenkette angegeben werden, muss diese in doppelten ' eingeklammert werden. Innerhalb der Zeichenkette werden die doppelten ' zu einfachen ' geändert. Hinter between muss in eckigen Klammern ein Bereich angegeben werden.



Bei der Verwendung von Vergleichen ohne die Berücksichtigung der Groß-/Kleinschreibung kann nur dann durch den Server eine Vorauswahl getroffen werden, wenn ein entsprechender Schlüssel mit Groß-/Kleinwandlung vorhanden ist.

Beispiele einfacher Ausdrücke beliebigen Typs:

## Kontakt

- 'ffArtPrice < 100.0'
- 'ffArtPrice < lReferencePrice'
- 'faCstName =\*^ '\*GMBH\*''
- 'faPlz between ["50000","69999"]'

### Vorauswahlen

Bei der Durchführung einer Selektion wird immer versucht eine Vorauswahl der Datensätze zu treffen, die die Selektionskriterien erfüllen können. Dazu werden Schlüssel herangezogen.

Im Normalfall ermittelt CONZEPT 16 die möglichen Schlüssel selbständig, um die Menge der zu untersuchenden Datensätze einzuschränken und optimiert damit die Selektion schon bei der Übersetzung. Der Anwender hat trotzdem die Möglichkeit, die Verwendung von Vorauswahlen zu beeinflussen. Zu diesem Zweck wird die Verarbeitung von Vorauswahlen nachfolgend erläutert.

Ohne eine Vorauswahl müssen alle Datensätze der Ausgangsdatei auf die Abfragekriterien hin überprüft werden. Je mehr Datensätze in der Datei enthalten sind, desto mehr Zeit nimmt dies in Anspruch. Mittels Vorauswahlen wird die Anzahl der tatsächlich zu überprüfenden Sätze reduziert, um eine schnellere Verarbeitung zu erreichen. Dazu muss ein Schlüssel vorhanden sein, der das in dem Kriterium überprüfte Feld als erstes Schlüsselfeld besitzt. Es können auch Schlüssel verwendet werden, in denen das Datenbankfeld als zweites oder späteres Schlüsselfeld angegeben ist. Es muss dann zwar der komplette Schlüssel durchsucht, die einzelnen Datensätze aber nicht gelesen werden. Bei der Vorauswahl wird über einen oder mehrere Schlüssel eine Zwischenmenge gebildet, über deren Datensätze anschließend die Abfrage durchgeführt wird.

Da die Bildung der Zwischenmenge meist nur wenige Sekunden in Anspruch nimmt, verringert sich die Gesamtzeit der Selektion deutlich, wenn nur ein Teil aller Datensätze überprüft werden muss.

In einem Ausdruck können optional die für Vorauswahlen zu verwendenden Schlüssel spezifiziert werden. Zu diesem Zweck wird nach dem Vergleich die Schlüsselnummer in geschweiften Klammern angegeben.

```
'faCstName =*^ '*GmbH*' {3}'
```

Bei mehreren Kriterien muss die Angabe vor der Kombination (siehe unten) stehen.

```
'faCstName =*^ '*GmbH*' {3} and faPlz between ["60000","60999"] {5}'
```

Durch die Angabe eines leeren Werts kann die Vorauswahl unterdrückt werden. Dies ist in den Fällen sinnvoll, in denen der weitaus größte Teil der Datensätze der Datei selektiert werden soll. Die Benutzung von Vorauswahlen kann in diesem Fall zu einer längeren Selektionszeit führen.

```
'fiCstId > 100000 {}'
```

## Kontakt

Durch die Angabe eines Pluszeichens nach der Schlüsselnummer kann die Spezifikation "Nur Vorauswahl" gesetzt werden. Wird nur eine Vorauswahl durchgeführt, findet eine Überprüfung des eigentlichen Kriteriums nicht mehr statt. Dabei ist zu beachten, dass bei alphanumerischen Schlüsseln je nach verwendeten Schlüsselattributen (beispielsweise "ohne Sonderzeichen") eine abweichende Ergebnismenge entstehen kann.

```
'faCstName =*^ '*GmbH*' {3+}'
```



Vorauswahlen sind nicht mit Schlüsseln möglich, deren Felder SOUNDEX 1 oder SOUNDEX 2 aktiviert haben.

Mehrere Ausdrücke können mit entsprechenden Operatoren verbunden werden:

and

or

xor

not

```
'ffArtPrice < lReferencePrice and faPlz between[''50000'', ''69999'']'
```

Die Auswertung der Vergleichsoperatoren findet immer vor der Auswertung der hier genannten Operatoren statt. Die Auswertung der Operatoren findet in der Reihenfolge von links nach rechts statt. Der Ausdruck a and b or c kann also ein anderes Ergebnis haben als c or b and a. Soll hier ein Teil des Ausdrucks vor einem anderen Teil ausgewertet werden, muss geklammert werden. Der Ausdruck a and (b or c) hat das gleiche Ergebnis wie (c or b) and a.

Innerhalb des Query-Strings stehen auch Funktionen zum Zugriff auf verknüpfte Datensätze zur Verfügung:

LinkTotal() Anzahl aller verknüpften Datensätze.

LinkCount() Anzahl der verknüpften Sätze, die die Abfrage erfüllen.

LinkMin() Minimalwert unter den verknüpften Sätzen.

LinkMax() Maximalwert unter den verknüpften Sätzen.

LinkAvg() Durchschnittswert aller verknüpften Sätze

LinkSum() Summe des Feldes von den verknüpften Sätzen.

Den Funktionen LinkTotal() und LinkCount() wird ein Abfragenamen übergeben. Die Funktionen geben ein ganzzahliges Ergebnis zurück.

Den Funktionen LinkMin() und LinkMax() können alle Feldtypen übergeben werden.

Der Rückgabewert entspricht dem übergebenem Typ.

Den Funktionen LinkAvg() und LinkSum() können nur numerische Feldtypen (word, int, bigint, float oder decimal) übergeben werden. Der Rückgabewert entspricht dem übergebenem Typ.

## Erstellen von logischen Ausdrücken in der Programmierung

Hier müssen deutlich zwei Ebenen unterschieden werden. Einerseits die Ebene des logischen Ausdrucks, der lediglich aus den hier vorgestellten Komponenten bestehen



## Kontakt

kann und andererseits die Ebene der Programmierung, die eine Vielzahl von Funktionen zur Verfügung stellt, mit denen eine Zeichenkette (der logische Ausdruck) zusammengefügt werden kann.

Um eine dynamische Selektion zu erstellen, wird zunächst ein Deskriptor für eine neue Selektion benötigt, anschließend kann eine Abfrage angegeben werden.

```
tHdlSel # SelCreate(tblCstCustomer, keyCstName); tHdlSel->SelDefQuery('', 'faCstLkz =^ ''D'' and f
```

Aus dem oben angegebenen Ausdruck ergibt sich folgende Abfrage: `faCstLkz =^ 'D' and faCstPlz between ['50000','69999']`. Die Feldinhalte der Felder `faCstLkz` und `faCstPlz` der Datensätze werden mit den angegebenen Konstanten verglichen.

Um die Selektion durchzuführen, muss sie in der Datenbank mit der Anweisung `SelStore()` gespeichert und übersetzt werden. Danach kann das Selektions-Objekt mit der Anweisung `SelOpen()` in einen Selektionspuffer umgewandelt, und anschließend die Selektion ausgeführt werden.

```
tErg # tHdlSel->SelStore('tmpSel', _SelLock); if (tErg = _ErrOk){ tHdlSel # tHdlSel->SelOpen();
```

Danach kann die erstellte Selektionsmenge ausgewertet und, wenn sie nicht weiter benötigt wird, auch geschlossen und gelöscht werden.

```
tHdlSel->SelClose(); SelDelete(tblCstCustomer, 'tmpSel');
```

Die Zeichenkette der Abfrage kann mit den Funktionen der Programmiersprache zusammengesetzt werden. Liegt zum Beispiel das Länderkennzeichen und der Postleitzahlbereich in Variablen vor, können diese entsprechend eingetragen werden.

```
tHdlSel->SelDefQuery('', 'faCstLkz =^ '' + tLkz + '' and faCstPlz between['' + tPlzFrom + '' ,
```



Die Variablen sind nicht Bestandteil der Abfrage. Der Inhalt der Variablen wird in die Zeichenkette eingefügt und ist dann Teil der Abfrage. So wie zum Beispiel `tLkz` immer außerhalb der Zeichenkette stehen muss, die die Abfrage darstellt, muss `between` immer innerhalb der Zeichenkette stehen.

Sind die Variablen zum Zeitpunkt der Ausführung des Befehls mit den Werten `tLkz = 'd'`, `tPlzFrom = '50000'` und `tPlzTo = '59999'` belegt, ergibt sich folgende Abfrage: `faCstLkz =^ 'd' and faCstPlz between['50000','59999']`.

Alles, was innerhalb der Zeichenkette steht, muss die Regeln für logische Ausdrücke befolgen, alles, was außerhalb der Zeichenkette steht, muss korrekter Programmcode sein. Da innerhalb der Zeichenkette ebenfalls das Zeichen `'` vorkommen muss, wird es doppelt angegeben, d. h. es wird innerhalb der Zeichenkette entsprechend umgewandelt.

```
tHdlSel->SelDefQuery('', 'faCstLkz =^ '' + tLkz + '' AND faCstPlz between['' + tPlzFrom + '' ,
```

Der Beginn und das Ende der Zeichenkette werden durch die roten `'` gekennzeichnet.

Aus der hier dargestellten Zeichenkette wird zum Beispiel folgende Abfrage:

```
faCstLkz =^ 'ch' and faCstPlz between['4000','4999']
```

## Kontakt

Unter der Voraussetzung, dass in den Variablen `tLkz` der Wert `'ch'` und in den Variablen `tPlzFrom` und `tPlzTo` die Werte `'4000'` und `'4999'` abgelegt sind. Durch die Angabe des Vergleichsoperator `=^` erfolgt der Vergleich unabhängig von der Groß-/Kleinschreibung.

### Zugriff auf Werte verknüpfter Dateien

Ergebnisse von der Abfrage verknüpfter Dateien können ebenfalls ausgewertet werden. Voraussetzung dafür ist eine entsprechende Verknüpfung in der Datenstruktur. Im folgenden wird davon ausgegangen, dass eine Verknüpfung von der Datei `tblCstCustomer` zu der Datei `tblOrdOrder` mit dem Namen `lnkCstOrd` existiert.

Das Anlegen der dynamischen Selektion erfolgt genauso, wie in dem vorhergehenden Beispiel.

```
tHdlSel # SelCreate(tblCstCustomer, keyCstName);
```

Anschließend wird eine verknüpfte Datei angehängt.

```
tHdlSel->SelAddLink('', tblOrdOrder, tblCstCustomer, lnkCstOrd, 'Order', 0);
```

Durch die Angabe des Namens können Abfragen auf beiden Dateien definiert werden. Mit dem Namen `"` wird die Abfrage auf der ersten Datei angesprochen. Im folgenden können die Abfragen definiert werden.

```
tHdlSel->SelDefQuery('Order', 'fdOrderDate between [1.1.2008,31.12.2008]'); tHdlSel->SelDefQuery('
```

Die Anweisung `LinkCount()` gibt alle verknüpften Datensätze zurück, die das Selektionskriterium erfüllt haben. Hier werden also alle Kunden erfasst, die im Jahr 2008 mindestens einen Auftrag erteilt haben. Die weitere Verarbeitung entspricht der weiter oben genannten Vorgehensweise.

## Kontakt

obj -> SelAddLink(alpha1, int2, int3,  
int4, alpha5[, int6])



Abfrage um Verknüpfung erweitern  
Deskriptor des

obj Selektionspuffers

**alpha1** Name der übergeordneten  
Abfrage

int2 Nummer der Ziel-Datei

int3 Nummer der Quell-Datei

int4 Nummer der Verknüpfung

alpha5 eigener Abfragenamen

Optionen (optional)

int6       SelResultSet Ergebnis in der  
Ergebnismenge  
speichern

Siehe Verwandte Befehle ,  
SelDefQuery(), SelAddResult()

Mit dieser Anweisung wird zu einer Selektion eine weitere Abfrage angehängt. Die Abfrage kann nur über eine Verknüpfung verbunden werden. Innerhalb einer Selektion können bis zu 16 Abfragen erstellt werden.

Als Objekt wird der Deskriptor, der von der Anweisung SelCreate() zurückgegeben wurde übergeben. In (alpha1) wird der Anker der neuen Abfrage angegeben. Hierbei handelt es sich entweder um den Namen einer anderen Abfrage oder um eine Leerstring. Der Leerstring entspricht dabei der ersten Abfrage der Selektion.

In den Parametern (int2), (int3) und (int4) wird die Verknüpfung angegeben. Die Parameter entsprechen dabei der Ziel-Datei, der Quell-Datei und der Nummer der Verknüpfung. Die Quell-Datei muss dabei die Ausgangsdatei der übergeordneten Abfrage sein. Existiert in der Quell-Datei keine entsprechende Verknüpfung oder stimmt die Ziel-Datei nicht überein, kommt es zu einem Laufzeitfehler.

Die Abfrage muss mit einem Namen versehen werden. Dieser Name gilt als Referenz für weitere Abfragen und zur Definition der Selektionskriterien. Der Name wird in (alpha5) angegeben. Er darf maximal 16 Zeichen lang sein und muss mit einem Buchstaben beginnen.

In (int6) kann die Konstante SelResultSet angegeben werden, um das Ergebnis der angehängte Abfrage als eine Ergebnismenge anzulegen. Zuvor muss mit der Anweisung SelAddResult() die Ergebnismenge angelegt worden sein. Wird der Parameter weggelassen oder 0 übergeben, wird das Ergebnis nicht in der Ergebnismenge abgelegt.

### Beispiele:

```
// Abfrage über Kunden -> Aufträge -> AuftragspositionentHdlSel # SelCreate(tblCstCustomer, keyCs
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid

## Kontakt

Der in (obj) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.

**ErrNoLink**

Die angegebene Verknüpfung ist nicht vorhanden.

**ErrLinkInvalid**

Die angegebene Verknüpfung stimmt nicht mit den anderen Definitionen überein.

**ErrIllegalOp**


Es wurden mehr als 16 Verknüpfungen angegeben.

**ErrValueInvalid**

Der eigene Abfragenname ist leer oder bereits vorhanden, oder der Name der übergeordneten Abfrage ist nicht vorhanden.

**ErrStringOverflow** Der eigene Abfragenname ist zu lang (maximal 16 Zeichen).

## Kontakt

obj -> SelAddResult(int1, int2, int3) | 

Selektion Ergebnismenge hinzufügen

obj Deskriptor des

Selektionspuffers

int1 Nummer der  
Ergebnisdatei

int2 Nummer der  
Ausgangsdatei

int3  
Verknüpfung  
Verwandte

Siehe Befehle,

SelAddLink()

Mit dieser Anweisung wird zu einer Selektion eine weitere Ergebnismenge definiert. In den Parametern wird die Ergebnisdatei, die Ausgangsdatei und die Verknüpfung angegeben. Anschließend kann bei der Anweisung SelAddLink() mit dem Parameter SelResultSet die verknüpften Datensätze in die Ergebnismenge aufgenommen werden.

### Beispiele:

```
// Abfrage über Kunden -> Aufträge -> AuftragspositionentHdlSel # SelCreate(tblCstCustomer, keyCs
```

### Mögliche Laufzeitfehler:

#### ErrHdlInvalid

Der in (obj) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.

#### ErrNoLink

Die angegebene Verknüpfung ist nicht vorhanden.

#### ErrLinkInvalid

Die angegebene Verknüpfung stimmt nicht mit den anderen Definitionen überein.

#### ErrIllegalOp

Es wurden mehr als 16 Verknüpfungen angegeben.

ErrSelUnknownOrInvalidLink In (int3) wurde eine falsche Verknüpfung angegeben. Die Datei ist bereits als Ergebnismenge definiert oder

#### ErrSelResultSet

entspricht der Hauptdatei.

## Kontakt

obj -> SelAddSortFld(int1, int2[, int3[,  
int4]])



Sortierung der Selektion definieren

obj    Deskriptor des Selektionspuffers

int1    Teildatensatznummer

int2    Feldnummer

         Schlüsselattribute (optional)

KeyFldAttrReverse            Umgekehrte

         Reihenfolge

KeyFldAttrUpperCase        Groß-/Kleinwandlung

KeyFldAttrUmlaut            Umlaute in  
         alphabetischer

int3                            Reihenfolge

KeyFldAttrSpecialChars Ohne Sonderzeichen

KeyFldAttrSoundex1        Wandlung nach

         Soundex 1

KeyFldAttrSoundex2        Wandlung nach

         Soundex 2

int4    Maximale Definitionslänge (optional)

Siehe Verwandte Befehle, SelCreate()

Diese Anweisung definiert ein Feld nach dem die Datensätze in der Selektionsmenge sortiert werden. Soll nach mehreren Feldern sortiert werden, muss die Anweisung mehrfach aufgerufen werden. Der Aufruf folgt der Priorität des Sortierkriteriums (zum Beispiel zuerst nach Datum und dann nach Uhrzeit). Es können bis zu acht Sortierfelder angegeben werden.



Die Gesamtlänge der Sortierfelder darf 240 Byte nicht überschreiten. Werden also zwei Felder vom Typ alpha mit je 250 Zeichen angegeben, wird bei voll besetztem ersten Feld das zweite Feld nicht mehr ausgewertet.

In (obj) wird der Deskriptor des Selektionspuffer übergeben, der von der Anweisung SelCreate() zurückgegeben wurde. In (int1) und (int2) wird die Nummer des Teildatensatzes und des Sortierfeldes angegeben.

Im Übergabeparameter (int3) können ein oder mehrere Schlüsselattribute angegeben werden.

Mehrere Schlüsselattribute werden mit Binär-ODER (!) kombiniert.

KeyFldAttrReverse            Umgekehrte Reihenfolge

KeyFldAttrUpperCase        Groß-/Kleinwandlung

KeyFldAttrUmlaut            Umlaute in alphabetischer Reihenfolge

KeyFldAttrSpecialChars Ohne Sonderzeichen

KeyFldAttrSoundex1        Wandlung nach Soundex 1

KeyFldAttrSoundex2        Wandlung nach Soundex 2

Schlüsselattribute, die nicht angewendet werden können (zum Beispiel

KeyFldAttrUpperCase bei einem ganzzahligen Feld) werden ignoriert.

Bei Feldern vom Typ alpha kann die Anzahl der für die Sortierung signifikanten Stellen in (int4) angegeben werden.

## Kontakt

### Beispiele:

```
// Sortierung über das NamensfeldtHdlSel # SelCreate(tblCstCustomer, 0);tHdlSel->SelAddSortFld(sb
```

### Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.
<u>ErrNoFld</u>	Es wurde ein nicht vorhandenes Feld angegeben.
<u>ErrIllegalOp</u>	Es wurde bereits eine Sortierung über Schlüssel angegeben, oder es wurden bereits acht Schlüsselfelder definiert.

**SelCreate(int1,  
int2[, int3]) :**



**handle**

**Selektion erzeugen**

**int1      Dateinummer**

**int2      Schlüsselnummer**

**int3      Ausgangsdatei (optional)  
            Deskriptor der**

Resultat handle

**neuen Selektion**

**Verwandte Befehle,**

**SelDefQuery(),**

Siehe

**SelStore(), SelClose(),**

**Dynamische Selektionen**

**(Blog)**

Mit diesem Befehl wird ein Selection-Objekt angelegt, in dem eine Selektion erstellt werden kann. In (int1) wird die Dateinummer übergeben. Soll die Selektion nach einem Schlüssel sortiert sein, kann in (int2) die Nummer des entsprechenden Schlüssels übergeben werden. Soll eine andere oder keine Sortierung verwendet werden, wird hier 0 übergeben.

Das hier erzeugte Selection-Objekt unterscheidet sich vom Selektionspuffer, der von dem Befehl SelOpen() angelegt wird. In dem Selektionspuffer kann eine Selektion durchgeführt werden. Mit dem Selection-Objekt kann eine Selektion erstellt werden.

In (int3) kann eine andere Ausgangsdatei angegeben werden. Dies ist dann sinnvoll, wenn über mindestens eine Verknüpfung zugegriffen werden soll und die Anzahl der Datensätze in dieser verknüpften Datei wesentlich geringer ist, als in der Datei, in der die Selektion angelegt wird (siehe Selektionen optimieren).

Der Rückgabewert ist der Deskriptor auf das Objekt. Konnte der Puffer nicht angelegt werden, wird ein Laufzeitfehler generiert. Wird der Deskriptor nicht mehr benötigt, muss er mit SelClose() wieder geschlossen werden.

**Beispiel**

```
// Selektion anlegen ohne SortierungtSel # SelCreate(tblCstCustomer, 0); tSel->SelDefQuery('', 'f
```

**Mögliche Laufzeitfehler:**

**ErrNoFile**

**Die angegebene Datei ist nicht vorhanden.**

**ErrNoKey**

**Der angegebene Schlüssel ist nicht vorhanden.**

**ErrOutOfMemory**

**Der Puffer für die neue Selektion konnte nicht angelegt werden.**



## Kontakt



obj -> SelDefQuery(alpha1,  
alpha2[, alpha3]) : int  
Abfrage der Selektion definieren

obj                    Deskriptor des  
                      Selektionspuffers  
alpha1                Name des Ankers  
alpha2                Abfrage  
                      Name der  
                      alpha3

                      Prozedur  
Resultat int Fehlerwert

Verwandte  
Siehe                Befehle,  
                      SelCreate()

Mit dieser Anweisung wird innerhalb einer neu angelegten Selektion eine Abfrage definiert. Als (obj) muss der Deskriptor übergeben werden, der von SelCreate() zurückgegeben wurde.

Als (alpha1) wird der Name eines Ankers angegeben. Der Anker wird benötigt, wenn die Abfrage einer Verknüpfung zugeordnet werden soll, die zuvor mit SelAddLink() in der Selektion erzeugt wurde. Als Anker muss der dort angegebene Name der Abfrage übergeben werden.

Als (alpha2) wird eine Zeichenkette mit der Abfrage-Bedingung übergeben. Die Zeichenkette muss einen logischen Ausdruck enthalten. Informationen zu logischen Ausdrücken befinden sich im Abschnitt Logische Ausdrücke in dynamischen Selektionen. Enthält die Abfrage eine Vorauswahl auf einen Schlüssel, bei dem ein Schlüsselfeld SOUNDEX 1 oder SOUNDEX 2 aktiviert hat, liefert der Befehl den Fehlerwert ErrSelInvalidKey.

In (alpha3) kann eine Prozedur oder Funktion angegeben werden, die für alle Datensätze, die das Abfragekriterium erfüllen durchgeführt wird. Der Rückgabewert der Prozedur oder Funktion bestimmt, ob der Datensatz in die Selektionsmenge aufgenommen wird (true) oder nicht (false).

Über den Rückgabewert kann überprüft werden, ob die Anweisung korrekt ausgeführt werden konnte:

Konstante	Fehler
<u>ErrOk</u>	Kein Fehler
<u>ErrSelUnknownField</u>	unbekanntes Feld
<u>ErrSelInvalidField</u>	ungültiges Feld
<u>ErrSelInvalidKey</u>	Vorauswahl mit ungültigem Schlüssel
<u>ErrSelUnknownOrInvalidLink</u>	unbekannte oder ungültige Verknüpfung
<u>ErrSelIllegalOperator</u>	unzulässiger Operator
<u>ErrSelQueryOverflow</u>	Abfrage zu lang
<u>ErrParserEndOfText</u>	Abfrage unvollständig
<u>ErrParserInvalidChar</u>	unerlaubtes Zeichen
<u>ErrParserInvalidConst</u>	ungültige Konstante

## Kontakt

<u>ErrParserWrongType</u>	falscher Typ angegeben
<u>ErrParserOutOfRange</u>	unzulässiger Wert
<u>ErrParserStringOverflow</u>	Zeichenkette zu lang
<u>ErrParserUnknownID</u>	unbekannter Anker
<u>ErrParserSyntax</u>	Syntaxfehler in der Abfrage
<u>ErrParserIllegalElement</u>	ungültiges Element
<u>ErrParserMissingParenthesis</u>	Klammer fehlt
<u>ErrParserMissingComma</u>	Komma fehlt

Neben dem Rückgabewert werden auch die Fehlereigenschaften des Selektionspuffers gesetzt. Die Eigenschaften können mit Hilfe des Deskriptors auf den Selektionspuffer abgefragt werden. Der Wert der Eigenschaft ErrCode entspricht dem Rückgabewert der Anweisung.

### ErrSource Selektionsbedingung

ErrCode Code der Fehlermeldung

ErrText Text der Fehlermeldung

ErrPos Position, wo der Fehler auftritt



Hochkommas (') in Zeichenketten müssen, mit weiteren Hochkommas, maskiert werden. Im Quelltext müssen somit 4 Hochkommas innerhalb einer Zeichenkette und 7 Hochkommas am Anfang oder am Ende einer Zeichenkette angegeben werden. Dabei gelten 3 Hochkommas als Markierung vom Anfang bzw. Ende der Zeichenkette. Siehe dazu in den Beispielen.

### Beispiele:

```
// Alle Kunden ermitteln, die eine Kundennummer kleiner als 1000 habentHdlSel # SelCreate(tblCstC  
// Alle Kunden, die in einem bestimmten Datumsbereich mindestens einen Auftrag hattentHdlSel # Se
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.



**LinkAvg(alpha1,alpha2) : var**

**Durchschnittswert aller verknüpfter Datensätze, die das  
Abfragekriterium erfüllen**

**alpha1    Name der Abfrage**

**alpha2    Name des Feldes**

**Resultat var Durchschnittswert**

**Siehe        SelDefQuery(),**

**LinkMin(), LinkMax()**

**Diese Anweisung kann innerhalb eines Abfragekriteriums der Anweisung SelDefQuery() angegeben werden. Als Übergabeparameter wird der Name der Abfrage und der Name des Feldes angegeben.**

**Die Namen werden der Funktion ohne umschließende ' übergeben. Der Funktion können nur numerische Feldtypen (word, int, bigint, float oder decimal) übergeben werden. Der Rückgabewert hat den Typ des übergebenen Feldes. Als Wert wird der Durchschnitt aller Werte der verknüpften Datensätze, die das Abfragekriterium erfüllen, zurückgegeben.**

## Kontakt

**LinkCount(alpha1) : int**



Anzahl der verknüpften Datensätze, die das Abfragekriterium erfüllen alpha1

Name der Abfrage

Resultat int Anzahl der verknüpften Datensätze, die das Abfragekriterium erfüllen siehe



SelDefQuery(), LinkTotal()

Diese Anweisung kann innerhalb eines Abfragekriteriums der Anweisung SelDefQuery() angegeben werden. Es liefert die Anzahl der verknüpften Datensätze zurück, die das Abfragekriterium erfüllen. Der Name der Abfrage wird als (alpha1) ohne umschließende ' übergeben.

**Beispiel:**

```
// Alle Kunden selektieren, die mindestens einen Auftrag im Jahr 2007 hattenHdlSel # SelCreate(t
```

## Kontakt

**LinkMax(alpha1,alpha2) :** var        Maximalwert unter  
den verknüpften Sätzen

**alpha1** Name der Abfrage

**alpha2** Name des Feldes

Resultat var Größter Wert in diesem Feld  —

SelDefQuery(), LinkMin(),

Siehe

LinkAvg()

Diese Anweisung kann innerhalb eines Abfragekriteriums der Anweisung SelDefQuery() angegeben werden. Als Übergabeparameter wird der Name der Abfrage und der Name des Feldes angegeben. Die Namen werden der Funktion ohne umschließende ' übergeben. In (alpha2) kann ein Feld beliebigen Feldtyps angegeben werden. Der Rückgabewert hat den gleichen Typ wie das übergebenen Feld und entspricht dem größten Wert des Feldes aus der Menge der verknüpften Datensätze, die dem Abfragekriterium entsprechen.

**LinkMin(alpha1, alpha2) : var**        Minimalwert unter  
den verknüpften Sätzen

**alpha1** Name der Abfrage

**alpha2** Name des Feldes

**Resultat var** Kleinster Wert in diesem Feld  Siehe

SelDefQuery(), LinkMax(), LinkAvg()

Diese Anweisung kann innerhalb eines Abfragekriteriums der Anweisung SelDefQuery() angegeben werden. Als Übergabeparameter wird der Name der Abfrage und der Name des Feldes angegeben. Die Namen werden der Funktion ohne umschließende ' übergeben. In (alpha2) kann ein Feld beliebigen Feldtyps angegeben werden. Der Rückgabewert hat den gleichen Typ wie das übergebenen Feld und entspricht dem kleinsten Wert des Feldes aus der Menge der verknüpften Datensätze, die dem Abfragekriterium entsprechen.

## Kontakt

**LinkSum(alpha1,alpha2) : var**



**Summe des Feldes in den verknüpften Datensätzen**

**alpha1**    Name der Abfrage

**alpha2**    Name des Feldes                      —

**Resultat var** Summe der Felder

Siehe        SelDefQuery(), LinkAvg()

Diese Anweisung kann innerhalb eines Abfragekriteriums der Anweisung SelDefQuery() angegeben werden. Als Übergabeparameter wird der Name der Abfrage und der Name des Feldes angegeben.

Die Namen werden der Funktion ohne umschließende ' übergeben. Der Funktion können nur numerische Feldtypen (word, int, bigint, float oder decimal) übergeben werden. Der Rückgabewert entspricht dem Typ des übergebenen Feldes.

Von allen verknüpften Datensätzen, die das Abfragekriterium erfüllen, wird das angegeben Feld summiert.








## Kontakt

**LinkTotal(alpha1) : int**              **Anzahl aller**  
verknüpften Datensätze  
**alpha1** Name der Abfrage  
**Resultat** int Anzahl aller verknüpften Datensätze  **Siehe**  
**SelDefQuery()**, **LinkCount()**

Diese Anweisung kann innerhalb eines Abfragekriteriums der Anweisung **SelDefQuery()** angegeben werden. Als Übergabeparameter wird der Name der Abfrage angegeben. Der Name wird der Funktion ohne umschließende ' übergeben. Der Rückgabewert entspricht der Anzahl aller verknüpfter Datensätze. Das Abfragekriterium wird nicht berücksichtigt.



## Kontakt

obj -> SelStore(alpha1, int2) : int  Selektion      

speichern und übersetzen

obj      Deskriptor eines  
         Selektionspuffers

alpha1    Name der Selektion

Optionen

SelLock            Selektion wird  
                         für exklusiven  
                         Zugriff

gesperrt

int2      SelSharedLock Selektion wird

für  
gemeinsamen  
Zugriff

gesperrt

SelUnlock           Selektion wird

nicht gesperrt

Resultat int Fehlerwert 

Siehe      Verwandte Befehle, SelCreate()

Mit diesem Befehl wird eine prozedural erstellte Selektion gespeichert und übersetzt. In (obj) wird der von der Anweisung SelCreate() zurückgegebene Deskriptor übergeben. In (alpha1) wird der Name angegeben, unter dem die Selektion gespeichert werden soll. Der Name darf maximal 20 Zeichen lang sein. In (int2) können Sperroptionen angegeben werden, die entscheiden, ob die Selektion nach der Speicherung gesperrt ist oder nicht. Soll die Selektion anschließend durchgeführt werden, muss sie mit SelLock gesperrt werden.

Nach dem Speichern und übersetzen der Selektion steht der Deskriptor des Selektionsobjekts immer noch zur Verfügung. Er kann mit SelOpen() in einen Selektionspuffer umgewandelt werden. Zum Schließen des Selektionsdeskriptors oder des -puffers muss SelClose() ausgeführt werden.

Der Rückgabewert der Anweisung gibt über den Erfolg oder Fehler sowohl beim Speichern als auch bei der Übersetzung aufschluss. Folgende Rückgabewerte können auftreten:

<u>rOk</u>	kein Fehler
<u>rLocked</u>	Die Selektion ist bereits von einem andern Benutzer gesperrt.
<u>rExists</u>	Die Selektion existiert bereits.
<u>rDeadlock</u>	Es ist eine Verklemmung aufgetreten.
<u>ErrGeneric</u>	Die Selektion ist defekt.
<u>ErrSelResultSet</u>	Die Selektion hat keine Ergebnismenge.
<u>ErrSelTableOverflow</u>	Beim Übersetzen und Speichern ist ein interner Fehler aufgetreten.
<u>ErrSelCodeOverflow</u>	Beim Übersetzen und Speichern ist ein interner Fehler aufgetreten.
<u>ErrSelNoQuery</u>	Im Query-String ist ein Verweis auf eine nicht existierende Abfrage enthalten.

## Kontakt

### Beispiel

```
tErg # tHdlSel->SelStore('CstSelZipcode', _SelLock); // Selektion speichern und übersetzen...tHd
```

### Mögliche Laufzeitfehler:

<u><b>ErrHdlInvalid</b></u>	Der in (obj) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.
<u><b>ErrValueInvalid</b></u>	In (alpha1) wurde ein ungültiger oder leerer Name angegeben.
<u><b>ErrStringOverflow</b></u>	Der in (alpha1) angegebene Name ist zu lang (maximal 20 Zeichen).

Konstanten für dynamische Selektionen

Konstanten für dynamische Selektionen

Befehle für

Siehe dynamische  
Selektionen

- SelLock
- SelSharedLock
- SelUnlock

\_SelLock

Selektion sperren

Wert 8 / 0x00000008

Verwandte

Befehle,

Siehe SelRead(),

SelStore(),

SelSharedLock

Option bei SelRead() und SelStore() durch die eine Selektion gesperrt werden kann.

Soll die Selektion nicht verändert, sondern nur lesend zugegriffen werden, reicht eine Sperrung mit der Option \_SelSharedLock aus.

**\_SelUnlock**

**Selektion entsperren**

Wert <sup>32</sup> /

**0x00000020**

**Verwandte**

Siehe Befehle,

SelRead(),

SelStore()

Option bei SelRead() und SelStore() durch die eine Selektion entsperrt werden kann.

obj -> SelClear() :



int

Selektion leeren

obj Selektionspuffer-Deskriptor

Leerungsergebnis

rOk

Leeren

erfolgreich

Resultat int rNoLock Selektion

nicht



gesperrt

rDeadlock Verklemmung

aufgetreten

Verwandte Befehle, SelOpen(),

Siehe

SelRead()

Mit dieser Funktion wird der Inhalt der Selektion im Puffer (obj) komplett gelöscht. Der Puffer wurde zuvor mit dem Befehl SelOpen() angelegt und die Selektion mit SelRead() sperrend gelesen. Nach SelClear() verfügt eine Selektion über eine leere Ergebnismenge. Dadurch kann beispielsweise die Ergebnismenge mit SelRecInsert() gefüllt werden. SelClear() funktioniert auch bei Selektionen, die noch nie durchgeführt wurden.

Das Löschen des Inhaltes von Selektionen die eine Wertmenge beinhalten, ist nicht möglich.

### Beispiel

```
tHdlSel # SelOpen();tErg # tHdlSel->SelRead(tblCstCustomer, _SelLock, 'SelPostcode');tErg # tHdlS
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

## Kontakt

obj -> SelClose()

Selektionspuffer entfernen

obj Selektionspuffer-Deskriptor

Verwandte Befehle,

**Siehe** SelOpen(), Beispiel

Mit dieser Funktion wird der durch SelOpen() generierte Selektionspuffer (obj) wieder entfernt.

### Beispiel

```
tHdlSel # SelOpen() ; ... tHdlSel->SelClose() ;
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

SelCopy(int1, alpha2,  
alpha3) : int Selektion  
kopieren int1  
Dateinummer alpha2  
Quellselektion alpha3  
Zielselektion



	<b>Kopierresultat</b>	
	<u>rOk</u>	Kopieren erfolgreich
	<u>rLocked</u>	Quellselektion (alpha2) gesperrt
	<u>rNoKey</u>	Quellselektion (alpha2) nicht vorhanden
	<u>rNoRec</u>	Datei (int1) nicht vorhanden
Resultat	<u>int</u> <u>rExists</u>	Zielselektion (alpha3) bereits vorhanden
	<u>rDeadlock</u>	Verklemmung ist aufgetreten
	<u>rFailed</u>	Zugriff auf Selektionsmenge nicht möglich (Clients < 5.8.11 erhalten <u>rNoRec</u> ).

Siehe Verwandte Befehle, SelDelete(),

## Beispiel

Mit dieser Funktion kann eine bestehende Selektion kopiert werden. Dabei werden nur die Abfragen, nicht aber eventuell vorhandene Ergebnismengen kopiert.

Parameter (int1) enthält dabei die Dateinummer, in der die zu kopierende Selektion definiert ist. Die zu kopierende Selektion wird in Parameter (alpha2) angegeben, in Parameter (alpha3) wird der Name der durch Kopieren neu erstellten Selektion bestimmt.

SelCopy() ermöglicht es, in einer Mehrbenutzerumgebung mehrere Benutzer gleichzeitig identische Selektionen durchführen zu lassen. Dazu wird eine Master-Selektion erstellt, die dann für den jeweiligen Benutzer temporär unter einem anderen Namen kopiert wird. Da bei der Namensvergabe für die Zielselektion auf Eindeutigkeit geachtet werden sollte, empfiehlt es sich, den Namen für die kopierte Selektion beispielsweise aus dem Benutzernamen (siehe UserName) und der Benutzer-ID (siehe UserNumber) zusammenzusetzen.

Da bei SelCopy() keine Ergebnismengen mitkopiert werden, muss die neu erstellte Selektion anschließend durchgeführt werden, um eine Ergebnismenge zu erhalten. Die Ergebnismenge der neuen Selektion kann dann mit den für eine Selektionsauswertung und -verarbeitung vorgesehenen Funktionen bearbeitet



werden.



**Der Rückgabewert des Befehls muss ausgewertet werden, um ein fehlerfreies Arbeiten der Prozedur zu kontrollieren.**

**Beispiel:**

```
// Name der Selektion erzeugentSelNew # 'TMP_SEL_' + UserInfo(_UserCurrent); // Selektion kopieren
```

SelDelete(int1,



alpha2) : int

Selektion löschen

int1      Dateinummer

alpha2    Selektionsname

    Löschresultat

rOk

    Löschen

    erfolgreich

rLocked

    Selektion

    (alpha2)

    gesperrt

Resultat int\_rNoRec

    Datei (int1)

    oder

    Selektion

    (alpha2) nicht

    vorhanden

rDeadlock Verklemmung

    aufgetreten

Verwandte Befehle, SelCopy(),


Siehe

Beispiel

Mit dieser Funktion kann eine bestehende Selektion komplett entfernt werden, einschließlich eventuell vorhandener Ergebnismengen. Dabei wird in Parameter (int1) die Nummer der Datei und in (alpha2) der Name der zu entfernenden Selektion angegeben.

### Beispiel

```
tErg # SelCopy(tblCstCustomer, 'SEL_PLZ', 'TMP_' + UserInfo(_UserCurrent));...tErg # SelDelete(tb
```

obj -> SelIgnore(int1, logic1) 

Selektionsvorauswahl ignorieren

obj Selektionspuffer-Deskriptor

int1 Schlüsselnummer

logic2 Ignorieren

aktivieren/deaktivieren

**Siehe**

SelOpen(), SelRun()

Mittels SelIgnore() können beim nächsten SelRun() Vorauswahlen der Selektion ignoriert werden. In (obj) wird der mit SelOpen() erzeugte Selektionspuffer angegeben. (int1) gibt die Nummer des Schlüssels an, bei dem keine Vorauswahlen durchgeführt werden sollen. Dadurch kann in einer Prozedur die Anzahl durchzuführender Vorauswahlen reduziert werden. Dies ist insbesondere mit dem Selektions-Parameter "Nur Vorauswahl" sinnvoll. Nach dem nächsten SelRun() sind alle durch SelIgnore() vorgenommenen Einschränkungen wieder aufgehoben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

ErrValueRange Schlüsselnummer (int1) kleiner 1 oder größer 255

obj -> SelInfo(int1[, int2]) : int  Numerische

Selektionsinformationen ermitteln/setzen

obj Selektionspuffer-Deskriptor

Informationstyp

SelFile Selektionsdatei

SelSort Selektionssortierung

int1 SelCount Werteanzahl

SelCountD Anzahl verschiedener

Werte

int2 Neue Schlüsselnummer (optional)

Resultat int Aktuelle Selektionsinformation

Verwandte Befehle, RecInfo(),

Siehe SelInfoAlpha(), SelInfoTime(),

SelRead(), SelValue(), SelOpen()

Mit dieser Funktion können verschiedene Informationen einer Selektion im Puffer (obj) ermittelt werden. Der Selektionspuffer muss zuvor mit SelOpen() angelegt und eine Selektion mit SelRead() gelesen werden.

Folgende Werte sind für (int1) definiert:

- SelFile

Das Resultat ist die Dateinummer, in der die Selektion definiert ist.

- SelSort

Das Resultat ist der aktuelle Sortierungsschlüssel der Selektion. Das Resultat ist 0, wenn die Sortierung nicht über einen Schlüssel erfolgt. Soll eine neue Sortierung gesetzt werden, wird in (int2) die Schlüsselnummer übergeben, nach der die Menge sortiert werden soll. Die Selektion muss danach erneut mit SelRun() durchgeführt werden.

- SelCount

Das Resultat ist die Anzahl von Werten in der Wertmenge. Die Selektion muss dabei eine Wertmenge enthalten.

- SelCountD

Das Resultat ist die Anzahl verschiedener Werte in der Wertmenge. Die Selektion muss dabei eine Wertmenge enthalten.



Die Anzahl der Datensätze einer Selektionsmenge kann über den Befehl RecInfo() ermittelt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

## Kontakt

obj -> SelInfoAlpha(int1[, alpha2]) : alpha  Alphanumerische

Selektionsinformationen ermitteln/setzen

obj Selektionspuffer-Deskriptor

Informationstyp

SelName Selektionsname

ermitteln

SelUser Letzten Benutzer

int1 ermitteln

SelMin Minimalwert ermitteln

SelMax Maximalwert ermitteln

SelRemarks Selektionsbemerkung

ermitteln/setzen

Neue Selektionsinformation

alpha2 (optional)

Resultat alpha Aktuelle Selektionsinformation

Verwandte Befehle, SelInfo(),

Siehe SelInfoDate(), SelInfoTime(),

SelRead(), SelValue()

Mit dieser Funktion können verschiedene Informationen einer Selektion im Puffer (obj) ermittelt (ein Argumente) oder gesetzt werden (zwei Argumente). Der Selektionspuffer muss zuvor mit SelOpen() angelegt und eine Selektion mit SelRead() gelesen werden.

Folgende Werte sind für (int1) definiert:

- SelName

Das Resultat ist der Name der Selektion. Dieser Wert kann nicht geändert werden.

- SelUser

Das Resultat ist der Name des Benutzers, der die Selektion zuletzt geändert hat. Dieser Wert kann nicht geändert werden.

- SelMin

Das Resultat ist der Minimalwert in der Wertemenge. Die Selektion muss dabei eine Wertemenge über ein alpha-Feld enthalten. Dieser Wert kann nicht geändert werden.

- SelMax

Das Resultat ist der Maximalwert in der Wertemenge. Die Selektion muss dabei eine Wertemenge über ein alpha-Feld enthalten. Dieser Wert kann nicht geändert werden.

- SelRemarks

Das Resultat ist die Bemerkung zu der Selektion.

Mögliche Laufzeitfehler:

## Kontakt

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

obj -> SelInfoDate(int1) : date  Selektionsinformationen

ermitteln (Datumswerte)

obj Selektionspuffer-Deskriptor

Informationstyp

SelModified Letztes

Änderungsdatum

int1 SelCreated Erstellungsdatum

SelExecuted Letztes

Durchführungsdatum

Resultat date Selektionsinformation

Verwandte Befehle, SelInfo(),

Siehe SelInfoAlpha(), SelInfoTime(),

SelRead(), SelValue(), Beispiel

Mit dieser Funktion können verschiedene Datumswerte einer Selektion im Puffer (obj) ermittelt werden. Der Selektionspuffer muss zuvor mit SelOpen() angelegt und eine Selektion mit SelRead() gelesen werden.

Folgende Werte sind für (int1) definiert:

- SelModified

Das Resultat ist das Datum der letzten Änderung der Selektion.

- SelCreated


Das Resultat ist das Datum der Erstellung der Selektion.

- SelExecuted

Das Resultat ist das Datum der letzten Durchführung der Selektion.

Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

obj -> SelInfoTime(int1) : time 

Selektionsinformationen ermitteln (Zeitwerte)

obj Selektionspuffer-Deskriptor

Informationstyp

SelModified Letztes

Änderungsuhrzeit

int1 SelCreated Erstellungsuhrzeit

SelExecuted Letztes

Durchführungsuhrzeit

Resultat time Selektionsinformation

Verwandte Befehle, SelInfo(),

Siehe SelInfoAlpha(), SelInfoDate(),

SelRead(), SelValue(), Beispiel

Mit dieser Funktion können verschiedene Zeitinformationen einer Selektion im Puffer (obj) ermittelt werden. Der Selektionspuffer muss zuvor mit SelOpen() angelegt und eine Selektion mit SelRead() gelesen werden.

Folgende Werte sind für (int1) definiert:

- SelModified

Das Resultat ist die Uhrzeit der letzten Änderung der Selektion.

- SelCreated

Das Resultat ist die Uhrzeit der Erstellung der Selektion.

- SelExecuted

Das Resultat ist die Uhrzeit der letzten Durchführung der Selektion.

Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig



**SelOpen([handle1]) : handle**  **Selektionspuffer**  
anlegen / umwandeln

**handle1** Selektionsdeskriptor (optional)

**Resultat** handle Selektionspuffer-Deskriptor

Verwandte Befehle, SelCreate(),

Siehe SelDefQuery(), SelStore(),  
SelClose(), SelRead(), Beispiel

Mit dieser Funktion kann ein Selektionspuffer angelegt oder eine prozedural erzeugte Selektion in einen Selektionspuffer umgewandelt werden. Wird ein neuer Selektionspuffer erzeugt, kann anschließend mit dem Befehl SelRead() eine Selektion gelesen werden.

Der hier erzeugte Selektionspuffer unterscheidet sich von dem mit SelCreate() erzeugten Selection-Objekt. Im Selektionspuffer kann eine Selektion durchgeführt werden. Mit dem Selection-Objekt kann eine Selektion erstellt werden.

Wird in (handle1) ein mit SelCreate() erzeugtes Selection-Objekt angegeben, wird dieses umgewandelt. Der Deskriptor kann anschließend wie ein Selektionspuffer mit gelesener Selektion behandelt werden.

**Beispiele:**

```
// Selektionspuffer anlegentHdlSel # SelOpen();tHdlSel->SelRead(tblCstCustomer, _SelLock, 'CstSel
```

**Mögliche Laufzeitfehler:**

ErrOutOfMemory

Der Selektionspuffer konnte wegen einem Speicherproblem nicht angelegt werden.

ErrHdlInvalid

Der in (handle) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.

obj ->

SelRead(int1,  
int2[, alpha3]) :



int

Selektion lesen

obj Selektionspuffer-Deskriptor

int1 Dateinummer

int2 Optionen (siehe Text)

alpha3 Selektionsname (optional)

Leserresultat

rOk Lesen  
erfolgreich

rLocked Selektion  
(alpha3)  
gesperrt  
rNoKey Selektion  
(alpha3)  
nicht  
vorhanden,  
nächste  
Selektion

Resultat int



gelesen  
rNoRec Keine  
weitere  
Selektion  
vorhanden  
rLastRec Selektion  
(alpha3)  
nicht  
vorhanden,  
letzte  
Selektion  
gelesen

Verwandte Befehle,

Siehe SelOpen(), DbSelection,  
Beispiel

Mit dieser Funktion wird die Selektion (alpha3) aus der Datei (int1) in den Puffer (obj) geladen. Der Puffer muss zuvor mit dem Befehl SelOpen() eingerichtet werden. Sofern die angegebene Selektion nicht vorhanden ist, wird die Selektion mit dem nächstgrößeren Namen geladen und rNoKey zurückgeliefert. Bei Verwendung von SelFirst, SelLast, SelNext und SelPrev kann auf den Selektionsnamen verzichtet werden.

In (int2) können folgende Optionen angegeben werden:

- SelFirst

Die erste Selektion der Datei wird geladen.

- SelLast

## Kontakt

Die letzte Selektion der Datei wird geladen.

- SelPrev

Die Selektion mit dem nächstkleineren Namen wird geladen. Sofern keine weitere Selektion in der Datei vorhanden ist, wird als Resultat rNoRec zurückgeliefert.

- SelNext

Die Selektion mit dem nächstgrößeren Namen wird geladen. Sofern keine weitere Selektion in der Datei vorhanden ist, wird als Resultat rNoRec zurückgeliefert.

- SelLock

Die gelesene Selektion wird gesperrt. Dies ist nur dann der Fall, wenn rOk zurückgeliefert wird (Selektion vorhanden). Ist die Selektion durch einen anderen Benutzer gesperrt, so ist das Resultat rLocked.

- SelSharedLock

Die gelesene Selektion wird gesperrt. Andere Benutzer können noch lesend auf die Selektion zugreifen. Mit dieser Sperr-Option kann die Selektion nur gelesen, nicht aber geändert werden. Soll die Selektion verändert werden, muss sie mit der Option SelLock gesperrt werden.

- SelUnlock

Die gelesene Selektion wird entsperrt.

- SelKeyMode

Alternativer Verarbeitungsmodus

- SelKeyUpdate

Schlüsselwerte in Selektion aktualisieren

Die Angabe von SelLock ist notwendig, wenn mit den Ergebnismengen der Selektion gearbeitet werden soll (siehe SelClear(), SelRecInsert(), SelRecDelete(), RecRead() und RecLink()).

Die gelesene Selektionsmenge kann in einem RecList-Objekt angezeigt werden, in dem der von SelOpen() zurückgegebene Deskriptor in die Eigenschaft DbSelection eingetragen wird.



Eine Selektion sollte immer dann verwendet werden, wenn ein deutlich kleinerer Teil der Datensätze das Selektionskriterium erfüllt. Als Alternative können auch Filter oder Verknüpfungen zum Einschränken der Anzahl der Datensätze verwendet werden.

### Beispiel

```
tHdlSel # SelOpen(); tErg # tHdlSel->SelRead(tblCstCustomer, _SelLock, 'SEL_PLZ'); tErg # tHdlSel->
```

Mögliche Laufzeitfehler:

## Kontakt

ErrHdlInvalid Selektions-Deskriptor (obj) ungültig

ErrStringOverflow Selektionsname (alpha3) zu lang

## Kontakt

obj -> SelRecDelete(int1) : int  Datensatz aus

Selektionsmenge löschen

obj Selektionspuffer-Deskriptor


int1 Dateinummer

Löschresultat

rOk Löschen erfolgreich

rNoKey Datensatz nicht vorhanden

rNoRec Selektionsmenge ist leer

Resultat int rNoLock Selektion (handle) nicht gesperrt 

rDeadlock Verklemmung aufgetreten

rFailed Zugriff auf temporären Baum nicht möglich  
(Clients < 5.8.11 erhalten rNoRec).

Siehe Verwandte Befehle, SelRead(), SelRecInsert()

Diese Funktion entfernt den aktuell im Hauptspeicher befindlichen Datensatz der Datei (int1) aus einer Ergebnismenge der Selektion im Puffer (obj).

Ist die Ergebnismenge der Selektion unsortiert, reicht es aus die Datensatz-ID des zu entfernenden Datensatzes zu setzen (siehe RecInfo()). Ist die Ergebnismenge sortiert, muss neben der Datensatz-ID der entsprechende Schlüsselwert gesetzt werden.

Dabei ist zu beachten, dass der Sortierungswert des Satzes in der Ergebnismenge mit dem tatsächlichen Sortierungswerten im Datensatz übereinstimmen muss, da sonst der Datensatz nicht gelöscht werden kann (beispielsweise ist die Selektionsmenge nach Namen sortiert, der Name im Datensatz hat sich mittlerweile aber verändert). SelRecDelete() kann sowohl bei der Hauptergebnismenge als auch bei verknüpften Ergebnismengen benutzt werden. Für die Hauptergebnismenge kann in (int1) auch 0 übergeben werden.

### Beispiel

```
// Ersten Datensatz aus einer Selektionsmenge löschtHdlSel # SelOpen();tErg # tHdlSel->SelRead(
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

ErrFileInvalid Dateinummer (int1) ungültig

## Kontakt

obj -> SelRecInsert(int1) : int         Datensatz in

Selektionsmenge einfügen

obj Selektionspuffer-Deskriptor

int1 Dateinummer

Einfügeresultat

rOk Einfügen erfolgreich

rExists Datensatz bereits

vorhanden

rNoLock Selektion (handle) nicht

Resultat int gesperrt

rDeadlock Verklemmung ist



aufgetreten

rFailed Zugriff auf temporären

Baum nicht möglich

(Clients < 5.8.11 erhalten

rNoRec).

Verwandte Befehle, SelRead(),

Siehe

SelRecDelete()

Diese Funktion fügt den aktuell im Hauptspeicher befindlichen Datensatz der Datei (int1) in eine Ergebnismenge der Selektion im Puffer (obj) ein. Dies ist sowohl bei der Hauptergebnismenge als auch bei verknüpften Ergebnismengen möglich. Für die Hauptergebnismenge kann auch 0 in (int1) übergeben werden.

Um einen Datensatz in eine Ergebnismenge aufzunehmen, muss die entsprechende Selektion von dem Benutzer gesperrt sein (siehe SelRead()).

### Beispiel

```
// Ersten Datensatz in eine Selektionsmenge aufnehmen tHdlSel # SelOpen(); tErg # tHdlSel->SelRead(
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

ErrFileInvalid Dateinummer (int1) ungültig

## Kontakt

obj -> SelRun(int1[,  
alpha2[, int3[, int4]]) : int  
Selektion durchführen



obj Selektionspuffer-Deskriptor

Optionen

SelDisplay

Statusanzeige

aktivieren

SelDisplayDelayed Statusanzeige

verzögert aktivieren

SelBreak

Abbruch möglich

SelWait

Statusanzeige nicht  
automatisch

SelServer

schließen

Selektion auf dem

SelServerAllFld

Server ausführen

Selektion auf dem

Server ausführen

und die gefüllten

Feldpuffer der

gesamten

Datenstruktur

übertragen

SelServerAutoFld

Selektion auf dem

Server ausführen

und die in den

Selektionselementen

verwendeten

Feldpuffer

übertragen

SelBase

Nachselektion

SelUnion

Vereinigungsmenge

bilden

SelInter

Schnittmenge bilden

SelMinus

Restmenge bilden

alpha2 Name der 2. Selektion (optional)

int3 Dateinummer der zweiten Selektion,  
sofern abweichend (optional)

int4 Satzlimit (optional)

Resultat int Durchführungsergebnis



rOK

Durchführung

erfolgreich

rLocked

2. Selektion

(alpha2) gesperrt

rNoKey

2. Selektion

(alpha2) kann

nicht verwendet

werden

## Kontakt

<u>rNoRec</u>	2. Selektion (alpha2) nicht vorhanden
<u>rNoLock</u>	Selektion (obj) nicht gesperrt
<u>rUserBreak</u>	Benutzerabbruch
<u>rDeadlock</u>	Verklemmung aufgetreten
<u>rLimitReached</u>	Limit überschritten
<u>ErrSelValueSet</u>	Kombination von zwei Selektionen mit Wertmenge
<u>ErrSelSortDiffer</u>	Kombination von zwei Selektionen mit abweichender Sortierung
<u>ErrSelSame</u>	Kombination von zwei identischen Selektionen
<u>ErrNoProcInfo</u>	Prozedur nach Abfrage nicht vorhanden
<u>ErrNoSub</u>	Funktion in der Prozedur nach Abfrage nicht vorhanden
<u>ErrCallOld</u>	Aufruf einer A- Prozedur
<u>ErrCodeUnknown</u>	Prozedurcode unbekannt
<u>ErrRights</u>	Keine ausreichenden Rechte zur Durchführung der Prozedur nach Abfrage
...	Laufzeitfehler der Prozedur nach Abfrage

Verwandte Befehle, [SelOpen\(\)](#),

Siehe [SelRead\(\)](#), [SelRecInsert\(\)](#),  
[SelRecDelete\(\)](#), [Beispiel](#), [Blog](#)

Diese Funktion führt die Selektion im Puffer (obj) durch. Der Puffer muss zuvor entweder mit [SelOpen\(\)](#) angelegt und eine Selektion mit [SelRead\(..., \[SelLock\]\(#\), ...\)](#) gelesen und gesperrt werden, oder es wird mit [SelCreate\(\)](#) eine neue Selektion angelegt und mit [SelOpen\(\)](#) in einen Selektionspuffer umgewandelt. Es ist zu



## Kontakt

beachten, dass SelRun() die Feldpuffer der Selektionsdatei verändert. Bei Angabe von \_SelBase, \_SelUnion, \_SelInter und \_SelMinus muss eine zweite Selektion in (alpha2) angegeben werden. Beide Selektionen müssen zuvor durchgeführt worden sein, um sie kombinieren zu können.

Über die Funktion RecRead() kann auf die Selektionsmenge zugegriffen werden. Für die Darstellung bzw. Ausgabe von Datensätzen einer Selektionsmenge in einem RecList- oder PrintDocRecord-Objekt steht die Eigenschaft DbSelection zur Verfügung.

Die Anzahl der selektierten Datensätze kann über die Funktion RecInfo() und bei einer Wertemengenselektion über die Funktion SelInfo() ermittelt werden.



Die Selektion darf nicht durchgeführt werden, wenn sie zur Zeit in einem RecList-Objekt angezeigt wird.

In (int1) können folgende Optionen angegeben werden:

- SelDisplay

Diese Option bewirkt eine Anzeige des Selektionsstatus während der Durchführung. Die Sprache des angezeigten Dialogs kann über die Eigenschaft LangDisplay gesteuert werden.

- SelDisplayDelayed

Diese Option entspricht der Option SelDisplay. Die Anzeige des Dialogs wird aber um zwei Sekunden verzögert. Ist zu diesem Zeitpunkt die Selektion bereits durchgeführt, erfolgt keine Anzeige. Die Verzögerungszeit kann in der Eigenschaft DisplayRaisingDelay eingestellt werden.

- SelBreak

Durch diese Option kann die Durchführung durch den Benutzer abgebrochen werden. SelBreak ist nur in Verbindung mit SelDisplay wirksam. Die Selektion kann ebenfalls durch eine Schaltfläche abgebrochen werden, deren Eigenschaft TypeButton auf den Wert WinBtnUserBreak gesetzt ist.

- SelWait

Diese Option gibt an, ob nach der Durchführung auf eine Benutzerbestätigung gewartet werden soll. SelWait ist nur in Verbindung mit SelDisplay wirksam.

- SelServer

Wird diese Option angegeben, wird die Selektion beim CONZEPT 16-Server durchgeführt. Werden Selektionen mit sehr vielen Zugriffen in die Datenbank beim Server ausgeführt, kann die Laufzeit dadurch erheblich verkürzt werden.

- SelServerAllFld

Wird diese Option angegeben, wird die Selektion beim CONZEPT 16-Server durchgeführt. Zusätzlich werden die gefüllten Feldpuffer der gesamten Datenstruktur übertragen.

- SelServerAutoFld

## Kontakt

Wird diese Option angegeben, wird die Selektion beim CONZEPT 16-Server durchgeführt. Zusätzlich werden die Feldpuffer der Felder übertragen, die in den Selektionselementen verwendet werden.



Wird die Selektion auf dem Server durchgeführt, können in einer Prozedur nach Abfrage nur Befehle verwendet werden, die auch vom Server ausgeführt werden können. Dies betrifft besonders den Zugriff auf Objekte der Oberfläche und den Aufruf von A-Prozeduren über die Anweisung CallOld().

- SelBase

Durch diese Option werden die in der Selektion (alpha2) enthaltenen Sätze als Selektionsgrundlage verwendet. Die Selektion (alpha2) bleibt unverändert.

- SelUnion

Mit dieser Option werden alle Sätze selektiert, die in der aktuellen Selektion oder in der Selektion (alpha2) enthalten sind (Vereinigungsmenge). Die resultierende Menge steht in der aktuellen Selektion. Die Selektion (alpha2) bleibt unverändert.

- SelInter

Mit dieser Option werden alle Sätze selektiert, die sowohl in der aktuellen Selektion als auch in der Selektion (alpha2) enthalten sind (Schnittmenge). Die resultierende Menge steht in der aktuellen Selektion. Die Selektion (alpha2) bleibt unverändert.

- SelMinus

Mit dieser Option werden alle Sätze selektiert, die in der aktuellen Selektion, aber nicht in der Selektion (alpha2) enthalten sind. Die resultierende Menge steht in der aktuellen Selektion. Die Selektion (alpha2) bleibt unverändert.

Bei einer Kombination von zwei Selektionen, die Wertmengen enthalten, kann die zweite Selektion auch aus einer anderen Tabelle stammen. Die Nummer der Tabelle wird in (int3) angegeben. Werden zwei Selektionsmengen miteinander kombiniert, müssen beide Selektionsmengen gleich sortiert sein. Die Selektionsmenge in (alpha2) darf auch nicht mit der durchzuführenden Selektion identisch sein. In diesen Fällen wird rNoKey zurückgegeben.

Die Konstanten SelInter, SelMinus und SelUnion können nicht mit den SelServer...-Konstanten kombiniert werden.

Im Parameter (int4) kann ein Satzlimit angegeben werden. Wird der Parameter weggelassen oder ein Wert < 0 angegeben, ist kein Limit gesetzt. Bei Werten >= 0 wird ein Limit gesetzt. Bei dem Wert 0 wird beispielsweise das Vorhandensein von Datensätzen, die dem Selektionskriterium entsprechen, geprüft. Die Selektion hört auf, wenn die Anzahl der gefundenen Datensätze der Hauptergebnismenge das Limit überschreitet. Entsprechen weitere Datensätze dem Selektionskriterium, wird als Resultat rLimitReached zurückgegeben, ansonsten rOK. Die Hauptergebnismenge enthält die gefundene Anzahl Datensätze, jedoch maximal bis zum Limit. Bei einer Kombination mit einer anderen Selektion ( SelUnion, SelInter oder SelMinus) wird

## Kontakt

das Limit ignoriert. Das Limit hat keinen Einfluss auf Wertmengenresultate.



Der Rückgabewert des Befehls muss ausgewertet werden, um ein fehlerfreies Arbeiten der Prozedur zu kontrollieren.

Besitzt die Selektion eine Prozedur nach Abfrage, wird bei einem Laufzeitfehler in dieser Prozedur die Selektion an dieser Stelle abgebrochen. Der Wert des Laufzeitfehlers wird als Ergebnis von SelRun() zurückgegeben.

Beispiele:

```
tHdlSel # SelRun();tErg # tHdlSel->SelRead(tblCstCustomer, _SelLock, 'SEL_PLZ');tErg # tHdlSel->S
```

Mögliche Laufzeitfehler:

<u>ErrNoFile</u>	Die gespeicherte Datei ist nicht vorhanden
<u>ErrHdlInvalid</u>	Der Selektionspuffer-Deskriptor (obj) ist ungültig
<u>ErrValueRange</u>	Die Option in (int1) ist ungültig
<u>ErrStringOverflow</u>	Der Selektionsname in (alpha2) ist zu lang

obj -> SelValue(int1) :



float

Selektionswerte ermitteln

obj Selektionspuffer-Deskriptor

Werttyp

SelSum Summe aller  
Werte

SelSumD Summe  
verschiedener  
Werte

int1 SelAvg Durchschnitt

aller Werte  
SelAvgD Durchschnitt  
verschiedener

Werte  
SelMin Minimalwert

SelMax Maximalwert

Resultat float Selektionswert

Verwandte Befehle,  
SelInfo(), SelInfoAlpha(),

Siehe

SelInfoDate(),  
SelInfoTime(), SelRead()

Mit dieser Funktion können verschiedene Informationen einer Selektion im Puffer (obj) ermittelt werden. Die Selektion muss dabei eine Wertmenge über ein numerisches Feld enthalten.

Folgende Werte sind für (int1) definiert:

- SelSum

Das Resultat ist die Summe aller Werte.

- SelSumD

Das Resultat ist die Summe aller verschiedenen Werte.

- SelAvg

Das Resultat ist der Durchschnitt aller Werte.

- SelAvgD

Das Resultat ist der Durchschnitt aller verschiedenen Werte.

- SelMin

Das Resultat ist der Minimalwert.

- SelMax

Das Resultat ist der Maximalwert.

Mögliche Laufzeitfehler:

## Kontakt

ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

Konstanten für Selektionsbefehle

Konstanten für Selektionsbefehle

Siehe Selektionsbefehle

- SelAvg
- SelAvgD
- SelBase
- SelBreak
- SelCount
- SelCountD
- SelCreated
- SelDisplay
- SelExecuted
- SelFile
- SelFirst
- SelInter
- SelKeyMode
- SelKeyUpdate
- SelLast
- SelLock
- SelMax
- SelMin
- SelMinus
- SelModified
- SelName
- SelNext
- SelPrev
- SelRemarks
- SelResultSet
- SelServer
- SelServerAllFld
- SelServerAutoFld
- SelSharedLock
- SelSort
- SelSum
- SelSumD
- SelUnion
- SelUnlock
- SelUser
- SelWait

## Kontakt

**\_SelAvg**

**Durchschnitt aller Werte**

**Wert 8**

**Verwandte**

Siehe Befehle,

SelValue(),

SelAvgD

**Option bei SelValue() durch die der Durchschnitt aller Werte ermittelt werden kann.**

**\_SelAvgD**

**Durchschnitt verschiedener Werte**

**Wert 9**

**Verwandte**

Siehe Befehle,

SelValue(),

SelAvg

**Option bei SelValue() durch die der Durchschnitt verschiedener Werte ermittelt werden kann.**



\_SelBase

Nachselektion

Wert 1

Verwandte

Siehe Befehle,

SelRun()

Option bei SelRun() durch die eine Nachselektion durchgeführt werden kann. Die Option kann mit den \_SelServer...-Konstanten kombiniert werden, um die Nachselektion auf dem Server auszuführen. Dies ist zum Beispiel sinnvoll, um eine Große Menge von Datensätzen neu zu sortieren.

## Kontakt

**\_SelBreak**

Abbruch möglich

Wert<sup>32/</sup>

**0x0020**

**Verwandte**

Siehe **Befehle**,

**SelRun()**

Option bei **SelRun()** durch die ein Benutzerabbruch ermöglicht werden kann.

Wird ein Selektionslauf abgebrochen, enthält die Selektionsmenge anschließend keine Datensätze. Ist eine Prozedur nach Abfrage angegeben, werden die Datensätze nicht durch diese Prozedur verändert.

## Kontakt

**\_SelCount**

Anzahl aller Werte

Wert 1

**Verwandte**

Siehe Befehle,

**SelInfo()**,

**SelCountD**

Option bei SelInfo() durch die die Anzahl aller Werte in der Wertmenge ermittelt werden kann.

Die Selektion muss dabei eine Wertmenge enthalten.

**\_SelCountD**

Anzahl verschiedener Werte

Wert 2

**Verwandte**

Siehe Befehle,

**SelInfo()**,

**\_SelCount**

Option bei SelInfo() durch die die Anzahl verschiedener Werte in der Wertmenge ermittelt werden kann.

Die Selektion muss dabei eine Wertmenge enthalten.

## Kontakt

**\_SelCreated**

Erstellungszeitpunkt ermitteln

Wert 1

### Verwandte

Siehe Befehle,

SelInfoDate(),

SelInfoTime()

Option bei SelInfoDate() und SelInfoTime() durch die das Datum und die Uhrzeit der Erstellung ermittelt werden kann.

**\_SelDisplay**

Statusanzeige aktivieren

Wert <sup>16</sup>/

**0x0010**

**Verwandte**

Siehe **Befehle**,

**SelRun()**

Option bei **SelRun()** durch die eine Statusanzeige während der Durchführung einer Selektion aktiviert werden kann.

**\_SelDisplayDelayed**

**Statusanzeige verzögert aktivieren**

Wert 1.040 /

**0x0410**

**Verwandte**

**Siehe Befehle,**

**SelRun()**

**Option bei SelRun() durch die eine Statusanzeige während der Durchführung einer Selektion aktiviert werden kann. Die Anzeige des Status erfolgt verzögert. Die Zeit, die gewartet wird, bevor eine Anzeige erscheint, ist in der Eigenschaft DisplayRaisingDelay des Selektions-Objekts (siehe SelOpen()) in Millisekunden angegeben.**

**\_SelExecuted**

**Letzten Durchführungszeitpunkt ermitteln**

**Wert 2**

**Verwandte**

Siehe **Befehle**,

**SelInfoDate()**,

**SelInfoTime()**

**Option bei SelInfoDate() und SelInfoTime() durch die das Datum und die Uhrzeit der letzten Durchführung ermittelt werden kann.**



## Kontakt

**\_SelFile**

**Selektionsdatei**

**Wert 3**

**Verwandte**

**Siehe Befehle,**

**SelInfo()**

**Option bei SelInfo() durch die die Datei einer Selektion ermittelt werden kann.**

**\_SelFirst**

**Erste Selektion lesen**

Wert <sup>1/</sup>

**0x00000001**

**Verwandte**

Siehe Befehle,

**SelRead()**,

**SelLast**

**Option bei SelRead() durch die die erste Selektion gelesen werden kann.**

## Kontakt

**\_SelInter**

**Schnittmenge bilden**

**Wert 3**

**Verwandte**

**Siehe Befehle,**

**SelRun()**

**Option bei SelRun() durch die eine Schnittmenge gebildet werden kann.**

**\_SelKeyMode**

Schlüsselwerte der Selektion entnehmen

Wert 268.435.456  
/ 0x10000000

Verwandte

Siehe Befehle,

SelRead()

Option bei SelRead() durch die die Schlüsselwerte einer Selektion entnommen werden können.

Bei der Verarbeitung von Selektionen per Prozedur stellt sich zum Teil das Problem, dass die in der Selektion benutzten Sortierfelder oder Schlüsselwerte sich zwischen Selektionsdurchführung und Prozedurverarbeitung (siehe Prozedur nach Abfrage) in den selektierten Sätzen verändert haben. Dadurch ergeben sich Probleme beim Zugriff auf die Selektionsmenge, da der Datensatzinhalt als Positionierungsgrundlage dient. Aus diesem Grund können bisher auch keine temporären Felder als Sortierelement benutzt werden, da diese nur während des Selektionslaufes generiert werden und daher nicht im Datensatz enthalten sind.

Mit der Option \_SelKeyMode kann eine alternative Verarbeitung eingeschaltet werden. Dabei werden nach dem Lesen des Satzes die Sortierfelder aus der Selektionsmenge in den Satz hinein entpackt. Dadurch ergeben sich keine Schwierigkeiten mehr bei der Positionierung oder bei der Verwendung von temporären Feldern.

Allerdings entspricht der Satz nicht mehr in jedem Fall dem in der Datenbank gespeicherten Zustand. Daher kann im alternativen Modus ein Satz beim Zugriff über die Selektionsmenge nicht gesperrt werden, um ein versehentliches Rückspeichern und Überschreiben der Originaldaten zu verhindern. Um bei der Selektionsverarbeitung im alternativen Modus Sätze zu verändern, müssen nach dem Lesen des Satzes die Sortierfelder zunächst zwischengespeichert und der Satz direkt (zum Beispiel über den 1.Schlüssel) gelesen und gesperrt werden. Danach kann der Satz geändert und rückgespeichert werden. Am Ende werden die Sortierfelder dann wiederhergestellt.



Die Gesamtlänge der Sortierfelder darf 240 Byte nicht überschreiten. Werden also zwei alpha-Felder mit je 250 Zeichen Länge als Sortierfelder angegeben, ist es möglich, dass nicht der vollständige Inhalt der Sortierfelder in den Datensatz entpackt wird, da nur maximal 240 Zeichen gespeichert werden. Das Feld wird dann einfach abgeschnitten. Andere Felder, die sich jenseits der 240 Byte-Grenze befinden, sind undefiniert.

### \_SelKeyUpdate

Schlüsselwerte in die Selektion übernehmen **Wert**  
536.870.912

/ 0x20000000

#### Verwandte

Siehe Befehle,

#### SelRead()

Option bei SelRead() durch die geänderte Schlüsselwerte einer Selektion beim Lesen von Datensätzen aktualisiert werden.

Werden Schlüsselwerte von Datensätzen, die Bestandteil einer Selektion sind, nachträglich verändert, führt dies dazu, dass die Schlüsselwerte in der Selektion von den Schlüsselwerten der Datensätze abweichen. Dies kann beim Lesen der Datensätze über RecRead() unter Angabe des Selektions-Deskriptors dazu führen, dass die Datensätze nicht in der korrekten Reihenfolge verarbeitet werden oder nicht alle Datensätze ermittelt werden können. Dies betrifft auch die Darstellung von Datensätzen in der RecList, wenn diese über Selektion gelesen werden.

Mit der Option \_SelKeyUpdate wird beim Lesen der Datensätze überprüft, ob der Schlüsselwert in der Selektion und der Schlüsselwert im Datensatz abweichen. Ist dies der Fall, wird der Schlüsselwert in der Selektion aktualisiert.

\_SelLast

Letzte Selektion lesen

Wert <sup>2</sup>/

0x00000002

Verwandte

Siehe Befehle,

SelRead(),

SelFirst

Option bei SelRead() durch die die letzte Selektion gelesen werden kann.

\_SelLock

Selektion sperren

Wert 8 / 0x00000008

Verwandte

Befehle,

Siehe SelRead(),

SelStore(),

SelSharedLock

Option bei SelRead() und SelStore() durch die eine Selektion gesperrt werden kann.

Soll die Selektion nicht verändert, sondern nur lesend zugegriffen werden, reicht eine Sperrung mit der Option \_SelSharedLock aus.

**\_SelMax**

**Maximalwert ermitteln**

**Wert 5**

**Verwandte**

**Befehle,**

**Siehe SelInfoAlpha(),**

**SelValue(),**

**SelMin**

**Option bei SelInfoAlpha() und SelValue() durch die der Maximalwert der Wertmenge ermittelt werden kann.**



\_SelMin

Minimalwert ermitteln

Wert 4

Verwandte

Befehle,

Siehe SelInfoAlpha(),

SelValue(),

SelMax

Option bei SelInfoAlpha() und SelValue() durch die der Minimalwert der Wertmenge ermittelt werden kann.

\_SelMinus

Restmenge bilden

Wert 4

Verwandte

Siehe Befehle,

SelRun()

Option bei SelRun() durch die eine Restmenge gebildet werden kann.

## Kontakt

**\_SelModified**

**Letzten Änderungszeitpunkt ermitteln**

**Wert 0**

### **Verwandte**

Siehe **Befehle**,

**SelInfoDate()**,

**SelInfoTime()**

**Option bei SelInfoDate() und SelInfoTime() durch die das Datum und die Uhrzeit der letzten Änderung ermittelt werden kann.**

## Kontakt

**\_SelName**

Selektionsname ermitteln

Wert 1

**Verwandte**

Siehe **Befehle**,

**SelInfoAlpha()**

Option bei **SelInfoAlpha()** durch die der Name einer Selektion ermittelt werden kann.

## Kontakt

\_SelNext

Nächste Selektion lesen

Wert <sup>4</sup>/

**0x00000004**

Verwandte

Siehe Befehle,

SelRead(),

SelPrev

Option bei SelRead() durch die die nächste Selektion gelesen werden kann.

**\_SelPrev**

**Vorherige Selektion lesen**

Wert <sup>3/</sup>

**0x00000003**

**Verwandte**

Siehe **Befehle**,

**SelRead()**,

**SelNext**

**Option bei SelRead() durch die die vorherige Selektion gelesen werden kann.**

## Kontakt

\_SelRemarks

Selektionsbemerkung ermitteln/setzen

Wert 6

Verwandte

Siehe Befehle,

SelInfoAlpha()

Option bei SelInfoAlpha() durch die die Bemerkung zu einer Selektion ermittelt/gesetzt werden kann.

## Kontakt

**\_SelResultSet**

**Resultat in der Ergebnismenge speichern**

**Wert 1**

**Siehe SelAddLink()**

**Wird bei der Anweisung SelAddLink() die Konstante \_SelResultSet angegeben, werden die verknüpften Datensätze, die das Selektionskriterium erfüllen, mit in der Ergebnismenge gespeichert.**

**Dies ist wichtig, wenn auch bei den verknüpften Datensätzen nur auf die Datensätze zugegriffen werden soll, die das Selektionskriterium erfüllen.**



**\_SelServer**

**Selektion auf dem Server ausführen**

Wert <sup>128 /</sup>

**0x0080**

**Verwandte**

Siehe **Befehle**,

**SelRun()**

Option bei **SelRun()**, durch die eine Selektion beim Server durchgeführt wird.

Selektionen können gerade bei vielen Zugriffen auf den Datenbestand mit dieser

Option beschleunigt werden.



Durch die Ausführung der Selektion beim Server, findet die Selektion in einem anderen Benutzerkontext statt. Der Benutzerkontext erbt die Benutzerrechte des ausführenden Benutzers, aber nicht den Inhalt der Feldpuffer. Falls in den Selektionselementen vorbelegte Feldpuffer zur Ermittlung der Selektionsmenge verwendet werden, muss anstelle von **\_SelServer** die Option **SelServerAutoFld** benutzt werden. Wird in der Selektion eine Prozedur nach Abfrage oder ein Ausdruck verwendet, in denen auf Feldinhalte zugegriffen wird, muss die Option **SelServerAllFld** verwendet werden.



In einer Prozedur nach Abfrage dürfen nur Befehle verwendet werden, die auch vom Server ausgeführt werden können. Dies betrifft besonders den Zugriff auf Objekte der Oberfläche und den Aufruf von A- Prozedur über die Anweisung **CallOld()**.

## Kontakt

**\_SelServerAllFld**

Ausführen der Selektion beim Server und Übertragen der Feldpuffer der gesamten Datenstruktur zum Server

**Wert** <sup>512</sup> /  
0x0200

Verwandte

Siehe Befehle,

SelRun()

Option bei SelRun(), durch die eine Selektion beim Server ausgeführt wird. Bei der Angabe dieses Parameters werden zusätzlich alle lokalen Feldinhalte der gesamten Datensatruktur zum Server übertragen. Leere Feldpuffer werden dabei ignoriert und nicht übertragen.



Durch diese Option können auch Selektionen auf dem Server ausgeführt werden, die beliebige lokale Feldpufferinhalte in Ausdrücken oder der Prozedur nach Abfrage verwenden. Große Datenstrukturen können bei der Initialisierung der Selektion eine geringfügige Verzögerung bedeuten.



In einer Prozedur nach Abfrage dürfen nur Befehle verwendet werden, die auch vom Server ausgeführt werden können. Dies betrifft besonders den Zugriff auf Objekte der Oberfläche und den Aufruf von A- Prozedur über die Anweisung CallOld().

**\_SelServerAutoFld**

Ausführen der Selektion beim Server und Übertragen der Feldpuffer zum Server  
Wert <sup>256</sup> /

**0x0100**

**Verwandte**

Siehe **Befehle**,

**SelRun()**

Option bei **SelRun()**, durch die eine Selektion beim Server ausgeführt wird. Speziell bei der Angabe dieses Parameters werden zusätzlich alle lokalen Feldinhalte der Felder zum Server übertragen, die in den Selektionselementen verwendet werden.



Feldinhalte der Felder, welche die Selektion in Ausdrücken oder der Prozedur nach Abfrage verwendet, können nicht automatisch zum Server übertragen werden. In diesen Fällen muss die Option **\_SelServerAllFld** verwendet werden.



In einer Prozedur nach Abfrage dürfen nur Befehle verwendet werden, die auch vom Server ausgeführt werden können. Dies betrifft besonders den Zugriff auf Objekte der Oberfläche und den Aufruf von A- Prozedur über die Anweisung **CallOld()**.

\_SelSharedLock

Selektion mehrfach sperren

Wert <sup>48</sup> /

0x00000030

Verwandte

Siehe Befehle,

SelRead(),

SelLock

Option bei SelRead() durch die eine Selektion mehrfach gesperrt werden kann.

Die gelesene Selektion wird gemeinsam mit anderen Benutzern gesperrt. Im Gegensatz zur Option \_SelLock können von anderen Benutzern weitere Sperren mit dieser Option eingerichtet werden. Eine mit \_SelSharedLock gesperrte Selektion kann von einem anderen Benutzer nicht mit der Option \_SelLock gesperrt werden, bis die letzte gemeinsame Sperre aufgehoben wurde.

Der Benutzer, der die gemeinsame Sperre eingerichtet hat, kann die Selektion nur dann mit \_SelLock sperren, wenn in der Zwischenzeit kein anderer Benutzer eine gemeinsame Sperre eingerichtet hat. Mit dieser Option können mehrere Benutzer eine Selektion öffnen. Diese Sperre erlaubt nur das Lesen der Selektion, nicht das Ändern.

Soll die Selektion geändert werden, muss eine Sperrung mit der Option \_SelLock erfolgen.

Die gemeinsame Sperre kann mit der Option \_SelUnlock aufgehoben werden.

## Kontakt

**\_SelSort**

**Selektionssortierung**

**Wert 0**

**Verwandte**

**Siehe Befehle,**

**SelInfo()**

**Option bei SelInfo() durch die die Sortierung einer Selektion ermittelt/gesetzt werden kann.**

**Die Sortierung der Selektionsmenge wird neu gesetzt oder abgefragt. Soll eine neue Sortierung gesetzt werden, wird in (int2) SelInfo() die Schlüsselnummer übergeben, nach dem die Menge sortiert werden soll. Die Selektion muss danach erneut mit SelRun() durchgeführt werden.**

\_SelSum

Summe aller Werte

Wert 6

Verwandte

Siehe Befehle,

SelValue(),

SelSumD

Option bei SelValue() durch die die Summe aller Werte ermittelt werden kann.

\_SelSumD

Summe verschiedener Werte

Wert 7

Verwandte

Siehe Befehle,

SelValue(),

SelSum

Option bei SelValue() durch die die Summe verschiedener Werte ermittelt werden kann.

**\_SelUnlock**

**Selektion entsperren**

Wert <sup>32</sup>/

**0x00000020**

**Verwandte**

Siehe Befehle,

SelRead(),

SelStore()

Option bei SelRead() und SelStore() durch die eine Selektion entsperrt werden kann.



## Kontakt

**\_SelUnion**

**Vereinigungsmenge bilden**

**Wert 2**

**Verwandte**

**Siehe Befehle,**

**SelRun()**

**Option bei SelRun() durch die eine Vereinigungsmenge gebildet werden kann.**

## Kontakt

\_SelUser

Letzten Benutzer ermitteln

Wert 2

Verwandte

Siehe Befehle,

SelInfoAlpha()

Option bei SelInfoAlpha() durch die der Benutzer, der eine Selektion zuletzt durchgeführt hat, (siehe SelRun()) ermittelt werden kann.

## Kontakt

\_SelWait

Statusanzeige nicht automatisch schließen

Wert <sup>64</sup> /

**0x0040**

Verwandte

Siehe Befehle,

SelRun()

Option bei SelRun() durch die ein automatisches Schließen der Statusanzeige nach der Durchführung einer Selektion verhindert werden kann.

## Textbefehle

Befehle zum Bearbeiten von Texten  
Siehe Befehlsgruppen,

Befehlsliste

### Befehle

- TextClear
- TextClose
- TextCopy
- TextCreate
- TextDelete
- TextInfo
- TextInfoAlpha
- TextInfoDate
- TextInfoTime
- TextLineRead
- TextLineWrite
- TextOpen
- TextRead
- TextRename
- TextSearch
- TextSearchRegEx
- TextWrite

### Konstanten

- TextANSI
- TextAuthRead
- TextAuthWrite
- TextClipboard
- TextCreated
- TextDb?
- TextEncrypted
- TextExtern
- TextFirst
- TextGroup
- TextLast
- TextLineDelete
- TextLineInsert
- TextLines
- TextLock
- TextModified
- TextName
- TextNext
- TextNoContents
- TextNoLineFeed
- TextOEM
- TextPrev
- TextPrivate
- TextProc
- TextRemarks

- TextRightRead
- TextRightWrite
- TextSearchCI
- TextSearchCIm
- TextSearchCount
- TextSearchLen
- TextSearchToken
- TextSharedLock
- TextSingleLock
- TextSize
- TextUnlock
- TextUserFirst
- TextUserLast
- TextUserPrivate

obj ->



**TextClear()**

Textpuffer leeren

obj    **Textpuffer-Deskriptor**

Verwandte Befehle

Siehe **TextOpen()**

Mit dieser Funktion wird Textkopf und Inhalt des Textpuffers (obj) geleert. Der Textpuffer muss zuvor mit **TextOpen()** angelegt werden.

Mögliche Laufzeitfehler:

**ErrHdlInvalid** Textpuffer-Deskriptor (obj) ungültig

obj -> TextClose() 

Textpuffer löschen

obj      Textpuffer-Deskriptor  
         Verwandte Befehle,

Siehe

TextOpen()

Mit dieser Funktion wird der mit TextOpen() angelegte Textpuffer (obj) gelöscht.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

**TextCopy(alpha1,  
alpha2, int3) : int**



**Text/Prozedur kopieren**

**alpha1**    Text-/Prozedurname

**alpha2**    Text-/Prozedurname (Kopie)

**Optionen**

**int3**      TextProc    Prozedur kopieren

TextDbas? Datenbankbereich

**Kopierresultat**

rOk            Kopieren

                  erfolgreich

rLocked        Text (alpha1)

                  gesperrt

rNoRec        Text (alpha1)

                  nicht

**Resultat int**    vorhanden 

rExists        Text bereits

                  vorhanden

rNoRights    Benutzerrechte

                  nicht

                  ausreichend

rDeadlock    Verklemmung

                  aufgetreten

**Verwandte Befehle, TextCreate(),**

Siehe

**TextDelete(), TextRename()**

Der interne Text (alpha1) wird unter dem Namen (alpha2) kopiert, sofern der Benutzer ausreichende Berechtigung besitzt. Der neue Text darf noch nicht vorhanden sein.

Folgende Optionen (int3) sind zulässig:

- **TextProc**

Die Prozedur (alpha1) wird unter dem Namen (alpha2) kopiert.

- **TextDbas?**

Der Text wird in einer anderen Datenbank kopiert. Die Datenbank wurde zuvor mit DbasConnect() mit einem Nummernbereich verbunden. Der Nummernbereich wird in der Option mit TextDbas2 bis TextDbas8 angegeben.



Die Kopie des Textes wird zu allen Elementgruppen hinzugefügt, bei denen der Originaltext enthalten ist. Somit erhält die Kopie die gleichen Benutzerberechtigungen. Ist dies nicht gewünscht, kann der Text mit TextRead() und TextWrite() kopiert werden.



Das Kopieren eines Textes von der Primär-Datenbank in eine verbundene Datenbank ist über die Funktion TextCopy() nicht möglich. Dazu kann die Funktion TextWrite() verwendet werden.

Mögliche Laufzeitfehler:



## Kontakt

ErrStringOverflow Textname (alpha1) oder (alpha2) zu lang

**TextCreate(alpha1,**



**int2) : int**

**Text/Prozedur anlegen**

**alpha1** Text-/Prozedurname

**Optionen**

**int2** TextLock Text sperren

TextProc Prozedur anlegen

TextDbas? Datenbankbereich

Anlegeresultat

rOk

Anlegen

erfolgreich

Resultat int rExists

Text bereits



vorhanden

rDeadlock Verklemmung

aufgetreten

Verwandte Befehle, TextCopy(),

Siehe

TextDelete()

Der interne Text (alpha1) wird angelegt.

Folgende Optionen (int2) sind zulässig:

- TextLock

Der Text wird direkt nach dem Anlegen gesperrt.

- TextProc


Der Text wird als Prozedur angelegt.

- TextDbas?

Der Text wird in einer anderen Datenbank angelegt. Die Datenbank wurde zuvor mit DbasConnect() mit einem Nummernbereich verbunden. Der Nummernbereich wird in der Option mit TextDbas2 bis TextDbas8 angegeben.

Mögliche Laufzeitfehler:

ErrMsgStringOverflow Textname (alpha1) zu lang

TextDelete(alpha1, int2) : int 

Text/Prozedur löschen

alpha1 Text-/Prozedurname

Optionen

0 Internen Text löschen

TextProc Prozedur löschen

TextDbas? Datenbankbereich

Löschresultat

rOk Löschen

erfolgreich

rLocked Text (alpha1)

gesperrt

rNoRec Text (alpha1)

Resultat int nicht 

vorhanden

rNoRights Benutzerrechte

nicht

ausreichend

rDeadlock Verklemmung

aufgetreten

Verwandte Befehle, TextCreate(),

Siehe

TextCopy()

Der interne Text (alpha1) wird gelöscht, sofern der Benutzer ausreichende Berechtigung besitzt.

Folgende Optionen (int2) sind zulässig:

- 0

Der interne Text (alpha1) wird gelöscht.

- TextProc

Die Prozedur (alpha1) wird gelöscht.

- TextDbas?

Der interne Text oder die Prozedur (bei Kombination mit TextProc) wird in einer anderen Datenbank gelöscht. Die Datenbank wurde zuvor mit DbConnect() mit einem Nummernbereich verbunden. Der Nummernbereich wird in der Option mit TextDbas2 bis TextDbas8 angegeben.

Mögliche Laufzeitfehler:

ErrStringOverflow Textname (alpha1) zu lang

## Kontakt

obj -> TextInfo(int1[, int2]) : int         Numerische Text-

---

obj        Textpuffer-Deskriptor  
int1       Informationstyp (siehe Text)  
            Neue Text-/Prozedurinformation  
int2

(optional)  
Resultat int Aktuelle Text-/Prozedurinformation  
          Verwandte Befehle, TextInfoAlpha(),

Siehe TextInfoDate(), TextInfoTime()

Mit dieser Funktion können verschiedene Informationen eines Textes im Textpuffer (obj) ermittelt (zwei Argumente) oder gesetzt werden (drei Argumente).

Folgende Informationen können nur gelesen werden:

- TextLines

Das Resultat ist die Anzahl der Textzeilen im Textpuffer. Ist kein Textpuffer vorhanden oder wurde kein Textinhalt geladen, so ist dies die im Textkopf eingetragene Zeilenanzahl.

- TextSize

Das Resultat ist die aktuelle Textgröße in Bytes. Die Textgröße ist nur bei internen Texten definiert und wird nur beim Speichern des Textinhalts erneuert.

- TextAuthRead

Verfügt der Benutzer über die Berechtigung den Text zu lesen ist das Resultat 1 andernfalls 0.

- TextAuthWrite

Verfügt der Benutzer über die Berechtigung den Text zu schreiben ist das Resultat 1 andernfalls 0.

- TextSearchCln

Liefert die Spaltenposition der über die Funktion TextSearch() und TextSearchRegEx() gefundenen Zeichenfolge.

- TextSearchLen

Liefert die Zeichenanzahl der über die Funktion TextSearch() und TextSearchRegEx() gefundenen Zeichenfolge.

- TextNoLineFeed

Beim Lesen werden die Textzeilen zu maximal 250 Zeichen zurückgeliefert. Über TextNoLineFeed kann festgestellt werden, ob die Zeile über einen festen (Resultat = 0) oder einen weichen Zeilenumbruch (Resultat = 1) verfügt. Bevor der Umbruch bestimmt werden kann, muss die Zeile mit der Funktion TextLineRead() gelesen worden sein.



Wird die Zeile beim Lesen gelöscht, wird für die nächste Zeile festgestellt, ob diese einen weichen Zeilenumbruch hat statt für die zuletzt gelesene

## Kontakt

Zeile.

Folgende Informationen können gelesen und gesetzt werden:

- TextRightRead

Das Resultat ist die Leseberechtigung des Textes. Dieser Wert kann nicht höher als die entsprechende Berechtigungsstufe des Benutzers gesetzt werden.

- TextRightWrite

Das Resultat ist die Schreibberechtigung des Textes. Dieser Wert kann nicht höher als die entsprechende Berechtigungsstufe des Benutzers gesetzt werden.

- TextPrivate

Das Resultat ist 1, wenn der Text als 'privat' markiert ist. Andernfalls ist das Resultat 0. Um einen Text als 'privat' zu markieren, wird in (int2) 1 übergeben, 0 um die Markierung zu entfernen.

- TextEncrypted

Das Resultat ist 1, wenn der Text als chiffriert gespeichert ist. Andernfalls ist das Resultat 0. Bei einer Veränderung wird diese erst beim nächsten Speichern des Textinhalts wirksam.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

## Kontakt

obj -> TextInfoAlpha(int1[,alpha2]) : alpha  Alphanumerische Text-  
/Prozedurinformationen ermitteln/setzen

obj	Textpuffer-Deskriptor	
	Informationstyp	
	<u>TextName</u>	Textname ermitteln
	<u>TextUserFirst</u>	Benutzer der
		Erstellung ermitteln
	<u>TextUserLast</u>	Benutzer der letzten
		Änderung ermitteln
int1	<u>TextUserPrivate</u>	Benutzer des privaten
		Textes ermitteln
	<u>TextGroup</u>	Textgruppe
		ermitteln/setzen
	<u>TextRemarks</u>	Textbemerkung
		ermitteln/setzen
		Neue Text-/Prozedurinformation

alpha2 (optional)

Resultat alpha Aktuelle Text-/Prozedurinformation

Siehe Verwandte Befehle, TextInfo(),

TextInfoDate(), TextInfoTime()

Mit dieser Funktion können verschiedene Informationen eines Textes im Textpuffer (obj) ermittelt (zwei Argumente) oder gesetzt werden (drei Argumente).

Die Informationen können nur von internen Texten ermittelt oder gesetzt werden. Wurde ein externer Text in den Textpuffer geladen, können diese Informationen weder gesetzt noch ermittelt werden.

Folgende Werte sind für (int1) definiert:

- TextName

Das Resultat ist der aktuelle Name des internen Textes. Dieser Wert kann nicht geändert werden. Der Name eines Textes kann maximal 20 Zeichen lang sein.

- TextUserFirst

Das Resultat ist der Name des Benutzers, der den Text angelegt hat. Dieser Wert kann nicht geändert werden.

- TextUserLast

Das Resultat ist der Name des Benutzers, der den Text zuletzt gespeichert hat. Dieser Wert kann nicht geändert werden.

- TextUserPrivate

Das Resultat ist der Name des Benutzers, auf den der Text als 'privat' geschützt ist. Dieser Wert kann nicht geändert werden.

- TextGroup

Das Resultat ist die Textgruppe des Textes. Der Name der Textgruppe kann maximal 20 Zeichen lang sein.

## Kontakt


- TextRemarks

Das Resultat stellt die Bemerkungen zum Text dar. Bermerkungen können bis zu 240 Zeichen umfassen.

Mögliche Laufzeitfehler:

ErrHdlInvalid      Textpuffer-Deskriptor (obj) ungültig

ErrStringOverflow Textname oder -bemerkung (alpha2) zu lang

obj -> TextInfoDate(int1) : date  Text-/Prozedurinformationen  
ermitteln (Datumswerte)

obj      Textpuffer-Deskriptor  
         Informationstyp  
         TextModified Letztes  
int1                      Änderungsdatum  
                            ermitteln

TextCreated Erstellungsdatum  
                            ermitteln

Resultat date Text-/Prozedurinformation  
            Verwandte Befehle, TextInfo(),

Siehe

TextInfoAlpha(), TextInfoTime()

Mit dieser Funktion können verschiedene Datumsinformationen eines Textes im Textpuffer (obj) ermittelt werden.

Die Informationen können nur von internen Texten ermittelt werden. Wurde ein externer Text in den Textpuffer geladen, können diese Informationen nicht ermittelt werden.

Folgende Werte sind für (int1) definiert:

- TextModified

Das Resultat ist das Datum der letzten Änderung des Textes.


- TextCreated

Das Resultat ist das Datum der Erstellung des Textes.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig



obj -> TextInfoTime(int1) : time  Text-  
/Prozedurinformationen ermitteln (Zeitwerte)

obj      Textpuffer-Deskriptor  
          Informationstyp  
          TextModified Letzte  
int1                                   Änderungsuhrzeit  
   ermitteln

TextCreated Erstellungsuhrzeit  
   ermitteln

Resultat time Text-/Prozedurinformation

Verwandte Befehle,  
Siehe    TextInfo(),TextInfoAlpha(),  
          TextInfoDate()

Mit dieser Funktion können verschiedene Zeitinformationen (mit Hundertstelsekunden) eines Textes im Textpuffer (obj) ermittelt werden.

Die Informationen können nur von internen Texten ermittelt werden. Wurde ein externer Text in den Textpuffer geladen, können diese Informationen nicht ermittelt werden.

Folgende Werte sind für (int1) definiert:

- TextModified  
  
Das Resultat ist die Uhrzeit der letzten Änderung des Textes.
- TextCreated  
  
Das Resultat ist die Uhrzeit der Erstellung des Textes.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

## Kontakt

obj -> TextLineRead(int1, int2) :



alpha

Text-/Prozedurzeile lesen/löschen

obj        Textpuffer-Deskriptor

int1        Zeilennummer

Optionen

int2        TextLineDelete Zeile

löschen

Resultat alpha        Zeileninhalt

Verwandte Befehle,

TextOpen(),

Siehe

TextLineWrite(),

TextInfo()

Mit dieser Funktion wird der Inhalt der Zeile (int1) des Textpuffers (obj) gelesen.

Folgende Optionen (int2) sind zulässig:

- TextLineDelete

Die gelesene Zeile wird nach dem Lesen gelöscht.

Die erste Zeile des Textes hat die Zeilennummer 1.

Beim Lesen werden die Textzeilen zu maximal 250 Zeichen zurückgeliefert. Über TextInfo() mit der Option TextNoLineFeed kann festgestellt werden, ob die aktuelle Zeile über ein festes oder ein weiches Zeilenende verfügt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

obj -> TextLineWrite(int1,  
alpha2, int3)



Text-/Prozedurzeile schreiben

obj      Textpuffer-Deskriptor

int1      Zeilennummer

alpha2 Zeileninhalt

Optionen

TextLineInsert Zeile einfügen

int3      TextNoLineFeed Weichen

Zeilenumbruch

setzen

Verwandte Befehle, TextOpen(),

Siehe TextLineRead(), TextInfo()

Mit dieser Funktion wird die Textzeile (alpha2) als Zeile (int1) in den Textpuffer (obj) geschrieben.

Folgende Optionen (int3) sind zulässig:

- TextLineInsert

Die Zeile wird eingefügt, ansonsten wird die bestehende Zeile überschrieben. Um eine Zeile an den Text anzuhängen, wird (int1) mit der Anzahl der Zeilen (siehe TextInfo()) plus 1 belegt.

- TextNoLineFeed

Der Textpuffer kann maximal 250 Zeichen pro Zeile verarbeiten. Soll eine Textzeile mit mehr als 250 Zeichen pro Zeile erzeugt werden, um diese in einem TextEdit-Objekt darzustellen, muss mit Hilfe der Option TextNoLineFeed ein weicher Zeilenumbruch definiert werden.

Beispiel:

Textzeile mit 300 Zeichen erzeugen.

```
// Schreiben der ersten 250 ZeichenHdlTxt->TextLineWrite(1, StrCut(tLine300, 1, 250), _TextLineI
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

**TextOpen(int1) :**



**handle**

**Textpuffer anlegen**

**int1**      **Textpuffergröße in KB**

**Resultat** **handle** **Textpuffer-Deskriptor**

**Verwandte Befehle,**

Siehe **TextRead()**, **TextWrite()**,  
**TextClear()**, **TextClose()**

Mit dieser Funktion wird ein Textpuffer angelegt, mit dem Texte oder Prozeduren verarbeitet werden können. Die Puffergröße muss im Bereich von 16 bis 512 KB liegen. Wird ein Wert außerhalb des Bereiches angegeben, wird er auf die nächstliegende Grenze vergrößert oder verkleinert.

Im Textpuffer können Texte verarbeitet werden, deren einzelne Zeilen länger als 250 Zeichen lang sind. Damit der Text kompatibel zum Textsystem der CONZEPT 16 Version 4.2 ist, werden solche Zeilen nach 250 Zeichen mit einem weichen Zeilenumbruch gespeichert, d. h. eine Zeile aus 400 Zeichen wird in eine Zeile mit 250 und in eine Zeile mit 150 Zeichen gespeichert. Die erste Zeile im gespeicherten Text hat im Unterschied zur zweiten kein festes Zeilenende sondern ein weiches Zeilenende. Beim Schreiben von Textzeilen in den Textpuffer mit dem Befehl **TextLineWrite()** kann ein weicher Zeilenumbruch durch die Option **TextNoLineFeed** angegeben werden.

Beim Lesen werden die Textzeilen zu maximal 250 Zeichen zurückgeliefert. Mit dem Befehl **TextInfo()** und der Option **TextNoLineFeed** kann festgestellt werden, ob die aktuelle Zeile über ein festes oder ein weiches Zeilenende verfügt.

Die Textgröße ist auf 128 MB beschränkt.

obj ->

TextRead(alpha1,



int2) : int

Text/Prozedur lesen

obj      Textpuffer-Deskriptor

alpha1    Textname

int2      Optionen (siehe Text)

    Leserresultat

rOk      Lesen

    erfolgreich

rLocked    Text

    (alpha1)

rNoKey    gesperrt

    Text

    (alpha1)

    nicht

    vorhanden,

    nächster

Resultat int     

    Text



    gelesen

rNoRec    Kein

    weiterer

    Text

    vorhanden

rLastRec Text

    (alpha1)

    nicht

    vorhanden,

    letzter Text

    gelesen

Verwandte Befehle,

Siehe

TextOpen(), TextWrite()

Mit dieser Funktion wird der Text (alpha1) in den Textpuffer (obj) geladen, sofern der Benutzer ausreichende Berechtigung besitzt.

Folgende Optionen (int2) sind zulässig:

- TextFirst

    Der erste Text wird gelesen.

- TextLast

    Der letzte Text wird gelesen.

- TextNext

    Der nächste Text wird gelesen.

- TextPrev

    Der vorherige Text wird gelesen.

- TextLock

Der gelesene Text wird gesperrt.

- TextSingleLock

Der gelesene Text wird einfach gesperrt.

- TextSharedLock

Der gelesene Text wird mehrfach gesperrt.

- TextUnlock

Der gelesene Text wird entsperrt.

- TextExtern

Der externe Text mit dem Namen (alpha1) wird geladen. Diese Option kann nur mit der Option TextOEM oder TextANSI kombiniert werden.

Als Rückgabewerte werden dann Konstanten aus dem Bereich der externen Dateioperationen zurückgegeben.

- TextOEM

Die Option wird nur beim Zugriff auf eine externe Datei ausgewertet. Die Datei wird mit dem OEM-Zeichensatz gelesen.

- TextANSI

Die Option wird nur beim Zugriff auf eine externe Datei ausgewertet. Die Datei wird mit dem ANSI-Zeichensatz gelesen.

- TextDb?

Die Operation bezieht sich auf einen Text in einer anderen Datenbank. Die Datenbank wurde zuvor mit DbConnect() mit einem Nummernbereich verbunden. Der Nummernbereich wird in der Option mit TextDb2 bis TextDb8 angegeben.

- TextProc

Die Prozedur (alpha1) wird gelesen.

- TextNoContents

Der Textinhalt wird nicht eingelesen. Auf die Kopfinformationen kann über die Funktion TextInfo() zugegriffen werden.

- TextClipboard

Der Inhalt der Windows-Zwischenablage wird in den Textpuffer übertragen.




Konnte ein Text nicht geladen werden, kann mit den Befehlen TextInfo(), TextInfoAlpha(), TextInfoDate() und TextInfoTime() trotzdem auf die Kopfinformationen eines Textes zugegriffen werden. Auf die Informationen welchen Textes zugegriffen wird, ergibt sich aus dem Rückgabewert von TextRead.

Mögliche Laufzeitfehler:

## Kontakt

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

ErrStringOverflow Textname (alpha1) zu lang

TextRename(alpha1, alpha2, ) : int Text/Prozedur

umbenennen

alpha1 Text-/Prozedurname

alpha2 Neuer Text-/Prozedurname

Optionen

TextProc Prozedur

int3

umbenennen

TextDb? Datenbankbereich

Löschresultat

rOk

Umbenennen

erfolgreich

rLocked

Text (alpha1)

gesperrt

rNoRec

Text (alpha1)

nicht

Resultat int

rExists

vorhanden

Text (alpha2)

bereits

vorhanden

rNoRights

Benutzerrechte

nicht

ausreichend

rDeadlock

Verklemmung

aufgetreten

Siehe Verwandte Befehle, TextCopy()

Der interne Text (alpha1) wird in (alpha2) umbenannt, sofern der Benutzer ausreichende Berechtigung besitzt.

Folgende Optionen (int3) sind zulässig:

- TextProc

Die Prozedur (alpha1) wird umbenannt.

- TextDb?

Der Text wird in einer anderen Datenbank umbenannt. Die Datenbank wurde zuvor mit DbConnect() mit einem Nummernbereich verbunden. Der Nummernbereich wird in der Option mit TextDb2 bis TextDb8 angegeben.

Mögliche Laufzeitfehler:

ErrStringOverflow Textname (alpha1) oder (alpha2) zu lang



## Kontakt

obj -> TextSearch(int1, int2, int3, alpha4[,  
alpha5[, int6]]) : int



Zeichenfolge in einem Text suchen / ersetzen

obj        Textpuffer-Deskriptor

int1       Anfangszeile

int2       Anfangsspalte

Optionen

int3       TextSearchCI        Groß-/Kleinwandlung  
            TextSearchCount Suchbegriffsfunde  
            TextSearchToken Begriffsorientierte  
                                Suche

alpha4    Suchbegriff

alpha5    Ersatzbegriff (optional)

int6       Ersetzungsanzahl (optional)

Suchresultat

Resultat int        Zeilennummer  
                        Suchbegriffsfunde

Ersetzungsvorgänge

Siehe       Verwandte Befehle, TextOpen(),  
            TextInfo(), TextSearchRegEx(),  
            WinRtfSearch()

Mit dieser Funktion wird der Textpuffer (obj) nach der Zeichenfolge (alpha4) durchsucht. In (int1) wird die Zeile und in (int2) die Spalte angegeben, ab welcher die Suche erfolgen soll. Resultat der Funktion ist die Zeilennummer in der der Suchbegriff zum ersten mal gefunden wurde. Falls der Begriff nicht gefunden wurde ist das Resultat 0.

Beispiel:

```
if (tTextHdl->TextSearch(1, 1, _TextSearchCI, 'CONCEPT 16') > 0){ // Begriff gefunden ...}
```

Mit der Funktion TextInfo() mit der Option TextSearchClm kann die Spaltenposition der gefundenen Zeichenfolge ermittelt werden.

Folgende Optionen (int3) sind zulässig:

- TextSearchCI

Bei der Suche wird die Groß/-Kleinschreibung nicht beachtet.

- TextSearchCount

Das Resultat ist die Anzahl der Suchbegriffsfunde.

- TextSearchToken

Die Suchergebnisse beschränken sich auf ganze Wörter.

Falls ein Ersatzbegriff (alpha5) angegeben ist, wird der Suchbegriff (alpha4) an jeder Stelle im Text durch den Ersatzbegriff ersetzt und die Funktion liefert die Anzahl der Ersetzungsvorgänge zurück.

## Kontakt

In (int6) kann die Anzahl der Ersetzungsvorgänge begrenzt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

## Kontakt

obj -> TextSearchRegEx(int1, int2, int3, alpha4) : int Zeichenfolge mit regulären Ausdrücken in einem Text suchen



obj       Textpuffer-Deskriptor

int1       Anfangszeile

int2       Anfangsspalte

int3       Optionen

TextSearchCI Groß-/Kleinwandlung

alpha4     Regulärer Ausdruck

Resultat int Suchresultat (Zeilennummer oder 0)

Siehe       Verwandte Befehle, TextOpen(),

TextInfo(), TextSearch()

Mit dieser Funktion wird der Textpuffer (obj) mit Hilfe des regulären Ausdrucks (alpha4) durchsucht. In (int1) wird die Zeile und in (int2) die Spalte angegeben, ab welcher die Suche erfolgen soll. Resultat der Funktion ist die Zeilennummer in der eine Entsprechung des regulären Ausdrucks zum ersten mal gefunden wurde. Falls der Begriff nicht gefunden wurde ist das Resultat 0.

### Beispiel:

```
// Nicht auskommentierte Funktionsaufrufe von "SysFnc:SelectNode" ermittelnif (tTextHdl->TextSear
```

Mit der Funktion TextInfo() mit der Option TextSearchCln kann die Spaltenposition und mit der Option TextSearchLen die Länge der gefundenen Zeichenfolge ermittelt werden.

Folgende Optionen (int3) sind zulässig:

- TextSearchCI

Bei der Suche wird die Groß-/Kleinschreibung nicht beachtet.



Der Befehl kann von der DLL-Schnittstelle nur ausgeführt werden, wenn der Eintrag pgx\_extended = 1 in der Konfigurationsdatei c16\_pgxsvc.cfg gesetzt wurde.

### Fehlerwerte:

Folgende Fehlerwerte können von dem Befehl zurückgegeben werden:

Fehlerwert	Bedeutung
<u>ErrRegExRuleSyntax</u>	Syntaxfehler im regulären Ausdruck
<u>ErrRegExBadEscapeSequence</u>	Nicht aufgelöste Escape-Sequenz im Ausdruck
<u>ErrRegExPropertySyntax</u>	Ungültige Unicode-Eigenschaft
<u>ErrRegExNotSupported</u>	Verwendung einer Funktion, die nicht unterstützt wird
<u>ErrRegExMismatchedParentheses</u>	Falsch verschachtelte Klammern im regulären Ausdruck
<u>ErrRegExNumberTooBig</u>	Dezimalzahl zu groß
<u>ErrRegExBadInterval</u>	Fehler im {min,max} Intervall

## Kontakt

ErrRegExMaxLtMin

ErrRegExInvalidBackRef

ErrRegExInvalidFlag

ErrRegExLookBehindLimit

ErrRegExSetContainsString

ErrRegExMissingCloseBracket

ErrRegExInvalidRange

ErrRegExStackOverflow

Im Intervall {min,max} ist max kleiner als min  
Rückbezug auf eine nicht vorhandene Referenz

Ungültiger Modus

Rückschau Ausdrücke müssen eine beschränkte  
maximale Länge haben

Reguläre Ausdrücke können keine UnicodeSets  
mit Zeichenketten beinhalten

Fehlende schließende Klammer in einem

Klammerausdruck

In einer Zeichenmenge [x-y] ist x größer als y

Stapelüberlauf in der Ablaufverfolgung des  
regulären Ausdrucks

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

## Kontakt

```
obj -> TextWrite(alpha1,  
int2) : int
```



## Text/Prozedur schreiben

obj      Textpuffer-Deskriptor


alpha1	Textname
--------	----------

**int2**      **Optionen (siehe Text)**

## Schreibresultat

Ok Schreiben erfolgreich

**\_rLocked      Text (alpha2) gesperrt**

**Resultat int\_rNoRights Benutzerberechtigung  nicht ausreichend**

## rDeadlock Verklemmung

**aufgetreten**

## Verwandte Befehle, `TextOpen()`,

**Siehe**

## TextRead()

**Mit dieser Funktion wird der Inhalt des Textpuffers (obj) als Text (alpha1) gesichert, sofern der Benutzer ausreichende Berechtigung besitzt.**

**Wurde der Textpuffer mit 0 KB Größe angelegt (siehe TextOpen()) wird nur der Textkopf ohne den Textinhalt gesichert.**



**Wird der Text in der Datenbank gespeichert ist zu beachten, dass der Textname (alpha1) maximal 20 Zeichen lang sein darf. Zusätzlich ist die Größe von internen Texten und Prozeduren auf ca 119 MB (nach Anzahl Blöcken) beschränkt. Wird dieses Limit überschritten, wird der Laufzeitfehler ErrLimitExceeded ausgelöst. Der bis zu diesem Limit geschriebene Textinhalt ist in diesem Fall weiterhin vorhanden.**

**Folgende Optionen (int2) sind zulässig:**

- TextLock

**Der geschriebene Text wird gesperrt.**

- **\_TextSingleLock**

**Der geschriebene Text wird einfach gesperrt.**

- TextSharedLock

**Der geschriebene Text wird mehrfach gesperrt.**

- **TextUnlock**

**Der geschriebene Text wird entsperrt.**

- TextExtern

**Der Text wird extern mit dem Namen (alpha1) gesichert.**

- TextOEM

**Die Option wird nur beim Zugriff auf eine externe Datei ausgewertet. Die Datei wird mit dem OEM-Zeichensatz geschrieben.**

- **TextANSI**

## Kontakt

Die Option wird nur beim Zugriff auf eine externe Datei ausgewertet. Die Datei wird mit dem ANSI-Zeichensatz geschrieben.

- TextDb?

Die Operation bezieht sich auf einen Text in einer anderen Datenbank. Die Datenbank wurde zuvor mit DbConnect() mit einem Nummernbereich verbunden. Der Nummernbereich wird in der Option mit TextDb2 bis TextDb8 angegeben.

- TextProc

Die Prozedur (alpha1) wird geschrieben. Die gespeicherten Lesezeichen aus der Prozedur bleiben erhalten.

- TextNoContents

Der Textinhalt wird nicht gesichert.

- TextClipboard

Der Inhalt des Textpuffers wird in die Windows-Zwischenablage übertragen.

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

ErrStringOverflow Textname (alpha1) zu lang

ErrLimitExceeded Das Limit für interne Texte und Prozeduren (ca. 119 MB) ist überschritten.

Konstanten für Textbefehle

Konstanten für Textbefehle

Siehe Textbefehle

- TextANSI
- TextAuthRead
- TextAuthWrite
- TextClipBoard
- TextCreated
- TextDbg?
- TextEncrypted
- TextExtern
- TextFirst
- TextGroup
- TextLast
- TextLineDelete
- TextLineInsert
- TextLines
- TextLock
- TextModified
- TextName
- TextNext
- TextNoContents
- TextNoLineFeed
- TextOEM
- TextPrev
- TextPrivate
- TextProc
- TextRemarks
- TextRightRead
- TextRightWrite
- TextSearchCI
- TextSearchCln
- TextSearchCount
- TextSearchLen
- TextSearchToken
- TextSharedLock
- TextSingleLock
- TextSize
- TextUnlock
- TextUserFirst
- TextUserLast
- TextUserPrivate

\_TextANSI

ANSI-Zeichensatz verwenden

Wert 1.048.576 /

**0x00100000**

Verwandte

Befehle,

Siehe TextRead(),

TextWrite(),

TextOEM

Option bei TextRead() und TextWrite() durch die der ANSI-Zeichensatz (Windows-Zeichensatz) verwendet werden kann.



## Kontakt

**\_TextAuthRead**

**Leserecht ermitteln**

**Wert 9**

**Verwandte**

**Siehe Befehle,**

**TextInfo()**

**Option bei TextInfo() durch die das Leserecht des Benutzers für einen Text ermittelt werden kann.**

## Kontakt

**\_TextAuthWrite**

Schreibrecht ermitteln

Wert 10

**Verwandte**

Siehe **Befehle**,

**TextInfo()**

Option bei **TextInfo()** durch die das Schreibrecht des Benutzers für einen Text ermittelt werden kann.

**\_TextClipBoard**

**Windows-Zwischenablage verwenden**

-2.147.483.648

**Wert / 0x80000000**

**Verwandte**

Siehe **Befehle**,

**TextRead()**,

**TextWrite()**

**Option bei TextRead() und TextWrite() durch die ein Text aus der Zwischenablage gelesen bzw. in die Zwischenablage geschrieben werden kann.**

**\_TextCreated**

Erstellungszeitpunkt ermitteln

Wert 1

**Textbefehle,**

Siehe **TextInfoDate()**,

**TextInfoTime()**

Option bei **TextInfoDate()** und **TextInfoTime()** durch die das Datum bzw. die Uhrzeit der Erstellung ermittelt werden kann.

## Kontakt

**\_TextDb?**

**Text in anderer Datenbank ansprechen**

Wert **0x01000000 -**

**0x07000000**

Siehe Textbefehle,

**DbConnect()**

Option bei **TextCreate()**, **TextCopy()**, **TextDelete()**, **TextRead()**, **TextRename()** und **TextWrite()** durch die ein Text in einer anderen Datenbank angesprochen werden kann.

Zuvor muss diese Datenbank mit dem Befehl **DbConnect()** verbunden werden. Der dabei angegebene Nummernbereich bestimmt mit welcher Option Texte dieser Datenbank angesprochen werden können:

**Nummernbereich 2 : \_TextDb2**

**Nummernbereich 3 : \_TextDb3**

**Nummernbereich 4 : \_TextDb4**

**Nummernbereich 5 : \_TextDb5**

**Nummernbereich 6 : \_TextDb6**

**Nummernbereich 7 : \_TextDb7**

**Nummernbereich 8 : \_TextDb8**

## Kontakt

**\_TextEncrypted**

Verschlüsselungsmarkierung ermitteln/setzen Wert 6

**Verwandte**

Siehe **Befehle**,

**TextInfo()**

Option bei **TextInfo()** durch die eine Verschlüsselungsmarkierung ermittelt bzw. gesetzt werden kann.

## Kontakt

\_TextExtern

Externen Text lesen/schreiben

Wert 536.870.912

/ 0x20000000

Textbefehle,

Siehe TextRead(),

TextWrite()

Option bei TextRead() und TextWrite() durch die ein externer Text gelesen bzw. geschrieben werden kann.

**\_TextFirst**

**Ersten Text lesen**

**Wert <sup>1/</sup>**

**0x00000001**

**Verwandte**

**Siehe Befehle,**

**TextRead(),**

**TextLast**

**Option bei TextRead() durch die der erste Text gelesen werden kann.**



## Kontakt

**\_TextGroup**

Textgruppe ermitteln/setzen

Wert 5

**Verwandte**

Siehe **Befehle**,

**TextInfoAlpha()**

Option bei **TextInfoAlpha()** durch die die Textgruppe eines Textes ermittelt/gesetzt werden kann.

**\_TextLast**

**Letzten Text lesen**

Wert <sup>2/</sup>

**0x00000002**

**Verwandte**

Siehe Befehle,

**TextRead()**,

**TextFirst**

Option bei TextRead() durch die der letzte Text gelesen werden kann.

**\_TextLineDelete**

**Textzeile löschen**

**Wert 1**

**Verwandte**

**Siehe Befehle,**

**TextLineRead()**

**Option bei TextLineRead() durch die eine Textzeile gelöscht werden kann.**

## Kontakt

**\_TextLineInsert**

**Textzeile einfügen**

**Wert 1**

**Verwandte**

**Siehe Befehle,**

**TextLineWrite()**

**Option bei TextLineWrite() durch die eine Textzeile eingefügt werden kann.**

**\_TextLines**

**Zeilenanzahl ermitteln**

**Wert 1**

**Verwandte**

**Siehe Befehle,**

**TextInfo()**

**Option bei TextInfo() durch die die Anzahl der Zeilen eines Textes ermittelt werden kann.**

**\_TextLock**

**Text sperren**

**Wert 8 / 0x00000008**

**Verwandte**

**Befehle,**

**TextRead(),**

**Siehe TextWrite(),**

**TextSingleLock,**

**TextSharedLock,**

**TextUnlock**

**Option bei TextRead() und TextWrite() durch die ein Text gesperrt werden kann.**

**\_TextModified**

**Letzten Änderungszeitpunkt ermitteln**

**Wert 0**

**Textbefehle,**

**Siehe TextInfoDate(),**

**TextInfoTime()**

**Option bei TextInfoDate() und TextInfoTime() durch die das Datum bzw. die Uhrzeit der letzten Änderung ermittelt werden kann.**

## Kontakt

**\_TextName**

Textname ermitteln

Wert 1

**Verwandte**

Siehe **Befehle**,

**TextInfoAlpha()**

Option bei **TextInfoAlpha()** durch die der Name des Textes ermittelt werden kann.



**\_TextNext**

**Nächsten Text lesen**

Wert <sup>4/</sup>

**0x00000004**

**Verwandte**

Siehe Befehle,

**TextRead()**,

**TextPrev**

Option bei **TextRead()** durch die der nächste Text gelesen werden kann.

**\_TextNoContents**

**Nur Textkopf lesen/schreiben**

**Wert 268.435.456**

**/ 0x10000000**

**Verwandte**

Siehe Befehle,

**TextRead()**,

**TextWrite()**

**Option bei TextRead() und TextWrite() durch die das Lesen bzw. Schreiben eines Textes auf den Textkopf beschränkt werden kann.**

**\_TextNoLineFeed**

Weichen Zeilenumbruch ermitteln/setzen

Wert 8

**Verwandte**

Siehe Befehle,

TextInfo(),

TextLineWrite()

Option bei TextInfo() und TextLineWrite() durch die ein weicher Zeilenumbruch ermittelt bzw. gesetzt werden kann.

**\_TextOEM**

**OEM-Zeichensatz verwenden**

Wert **0** /

**0x00000000**

**Verwandte**

**Befehle,**

Siehe **TextRead(),**

**TextWrite(),**

**TextANSI**

**Option bei TextRead() und TextWrite() durch die der OEM-Zeichensatz (PC-Zeichensatz) verwendet werden kann.**

**\_TextPrev**

**Vorherigen Text lesen**

Wert <sup>3/</sup>

**0x00000003**

**Verwandte**

Siehe **Befehle**,

**TextRead()**,

**TextNext**

Option bei **TextRead()** durch die der vorherige Text gelesen werden kann.

## Kontakt

**\_TextPrivate**

**Privat-Markierung ermitteln**

**Wert 5**

**Verwandte**

**Siehe Befehle,**

**TextInfo()**

**Option bei TextInfo() durch die eine Privat-Markierung ermittelt werden kann.**

\_TextProc

Prozedur ansprechen

Wert 1.073.741.824

/ 0x40000000

Textbefehle,

TextDelete(),

Siehe TextRead(),

TextRename(),

TextWrite()

Option bei TextDelete(), TextRead(), TextRename() und TextWrite() durch die eine Prozedur angesprochen werden kann.



Wird eine Prozedur mit TextWrite() unter einem anderen Namen gespeichert, werden die gespeicherten Lesezeichen aus der Zielprozedur verwendet.

## Kontakt

**\_TextRemarks**

Textbemerkung ermitteln/setzen

Wert 6

**Verwandte**

Siehe **Befehle**,

**TextInfoAlpha()**

Option bei **TextInfoAlpha()** durch die die Bemerkung eines Textes ermittelt/gesetzt werden kann.



## Kontakt

**\_TextRightRead**

**Leseberechtigung ermitteln**

**Wert 3**

**Verwandte**

**Siehe Befehle,**

**TextInfo()**

**Option bei TextInfo() durch die die Leseberechtigung des Benutzers für einen Text ermittelt werden kann.**

## Kontakt

**\_TextRightWrite**

Schreibberechtigung ermitteln

Wert 4

**Verwandte**

Siehe **Befehle**,

**TextInfo()**

Option bei **TextInfo()** durch die die Schreibberechtigung des Benutzers für einen Text ermittelt werden kann.

**\_TextSearchCI**

**Groß-/Kleinschreibung ignorieren**

**Wert 1 / 0x01**

**Verwandte**

**Siehe Befehle,**

**TextSearch()**

**Option bei TextSearch() und TextSearchRegEx() durch die die Groß-/Kleinschreibung beim Suchen ignoriert werden kann.**

## Kontakt

**\_TextSearchCln**

Spaltenposition eines Suchergebnisses ermitteln Wert 7

**Verwandte**

Siehe **Befehle**,

**TextInfo()**

Option bei **TextInfo()** durch die die Spaltenposition eines Suchergebnisses ermittelt werden kann.

**\_TextSearchCount**

**Suchbegriffsfunde**

**Wert 16 / 0x10**

**Verwandte**

**Siehe Befehle,**

**TextSearch()**

**Option bei TextSearch() durch die die Anzahl der gefundenen Suchergebnisse ermittelt werden kann.**

## Kontakt

**\_TextSearchLen**

Zeichenanzahl eines Suchergebnisses ermitteln Wert 11

**Verwandte**

Siehe **Befehle**,

**TextInfo()**

Option bei **TextInfo()** durch die die Zeichenanzahl eines Suchergebnisses ermittelt werden kann.

**\_TextSearchToken**

**Begriffsorientierte Suche**

**Wert 2 / 0x02**

**Verwandte**

**Siehe Befehle,**

**TextSearch()**

**Option bei TextSearch() durch die die Suchergebnisse auf ganze Wörter beschränkt werden kann.**

**Der Suchbegriff wird nur dann gefunden, wenn nach ihm ein Worttrennzeichen steht. Worttrennzeichen sind alle Zeichen mit Ausnahme von Buchstaben oder Zahlen. Das Zeilenende trennt ebenfalls Wörter voneinander.**

## Kontakt

**\_TextSharedLock**

**Text mehrfach sperren**

**Wert 48 / 0x00000030**

**Verwandte**

**Befehle,**

**TextRead(),**

**Siehe TextWrite(),**

**TextLock,**

**TextSingleLock,**

**TextUnlock**

**Option bei TextRead() und TextWrite() durch die der gelesene Text für andere Benutzer gesperrt werden kann.**

**Der gelesene Text wird gemeinsam mit anderen Benutzern gesperrt.**

**Im Gegensatz zur Option TextLock oder TextSingleLock können von anderen Benutzern weitere Sperren mit dieser Option eingerichtet werden. Ein mit \_TextSharedLock gesperrter Text kann von einem anderen Benutzer nicht mit der Option TextLock oder TextSingleLock zum Schreiben gesperrt werden, bis die letzte Sperre aufgehoben wurde.**

**Der Benutzer, der die gemeinsame Sperre eingerichtet hat, kann den Text nur dann mit TextLock sperren, wenn in der Zwischenzeit kein anderer Benutzer eine gemeinsame Sperre eingerichtet hat.**

**Mit dieser Option können mehrere Benutzer einen Text vor Veränderung schützen.**

**Der Text kann mit dieser Sperre nicht zurückgeschrieben werden.**

**Die gemeinsame Sperre wird mit der Option TextUnlock aufgehoben.**



**\_TextSingleLock**

**Text einfach sperren**

**Wert 40 / 0x00000028**

**Verwandte**

**Befehle,**

**TextRead(),**

**Siehe TextWrite(),**

**TextLock,**

**TextSharedLock,**

**TextUnlock**

**Option bei TextRead() und TextWrite() durch die ein Text über die Benutzer-ID gesperrt werden kann.**

**Mit dieser Option wird der gleiche Text für den gleichen Benutzer nur einmal gesperrt. Beim Versuch, denselben Text ein zweitesmal zu sperren, wird das Resultat \_rLocked zurückgeliefert.**

**\_TextSize**

Textgröße ermitteln

Wert 2

**Verwandte**

Siehe **Befehle**,

**TextInfo()**

Option bei **TextInfo()** durch die die Größe eines Textes in Bytes ermittelt werden kann.

**\_TextUnlock**

**Text entsperren**

**Wert 32 / 0x00000020**

**Verwandte**

**Befehle,**

**TextRead(),**

**Siehe TextWrite(),**

**TextLock,**

**TextSingleLock,**

**TextSharedLock**

**Option bei TextRead() und TextWrite() durch die ein Text entsperrt werden kann.**

## Kontakt

**\_TextUserFirst**

**Benutzer der Erstellung ermitteln**

**Wert 2**

**Verwandte**

**Siehe Befehle,**

**TextInfoAlpha()**

**Option bei TextInfoAlpha() durch die der Benutzer, der den Text erstellt, hat ermittelt werden kann.**

## Kontakt

**\_TextUserLast**

Benutzer der letzten Änderung ermitteln  
Wert 3

Verwandte

Siehe Befehle,

TextInfoAlpha()

Option bei TextInfoAlpha() durch die der Benutzer, der den Text zuletzt geändert hat, ermittelt werden kann.

## Kontakt

**\_TextUserPrivate**

**Benutzer des privaten Textes ermitteln**

**Wert 4**

**Verwandte**

**Siehe Befehle,**

**TextInfoAlpha()**

**Option bei TextInfoAlpha() durch die der Benutzer des privaten Textes ermittelt werden kann.**

## Kontakt

### Benutzerbefehle

Befehle für Datenbankbenutzer  
Siehe Befehlsgruppen,

#### Befehlsliste

Die Benutzerbefehle werden in zwei Gruppen aufgeteilt. Mit den Urm-Befehlen können die Benutzerrechte gesetzt und abgefragt, Benutzer bzw. Benutzergruppen angelegt und gelöscht werden. Informationen zu den Benutzern oder Benutzergruppen können über deren Eigenschaften abgefragt werden.

Die User-Befehle greifen auf die derzeit angemeldeten Benutzer zu. Der Zugriff auf die Eigenschaften der alten Benutzer-Verwaltung wird durch die \_UrmOldProp...-Eigenschaften und die \_UrmOldPerm...-Berechtigungen ermöglicht. Alle Veränderungen im Benutzersystem durch Urm-Befehle laufen außerhalb der Datenbanktransaktionen ab und werden somit nicht durch Transaktionsbefehle beeinflusst.

### Befehle

- UrmClose
- UrmCreate
- UrmDelete
- UrmOpen
- UrmPermElementGet
- UrmPermElementGetRaw
- UrmPermGet
- UrmPermGetRaw
- UrmPermLevelGet
- UrmPermLevelSet
- UrmPermSet
- UrmPropGet
- UrmPropSet
- UrmPropType
- UrmRead
  
- UserClear
- UserCreate
- UserDelete
- UserID
- UserInfo
- UserName
- UserNumber
- UserPassword

### Konstanten

- UrmAllow
- UrmDeny
- UrmFirst
- UrmIdePermCreate
- UrmIdePermDelete
- UrmIdePermModify

- UrmIdePermRead
- UrmLast
- UrmLock
- UrmNext
- UrmOldPermAccess
- UrmOldPermDelete
- UrmOldPermEntry
- UrmOldPermExecLists
- UrmOldPermExecSelections
- UrmOldPermExecTransfers
- UrmOldPermLink
- UrmOldPermListFormats
- UrmOldPermModify
- UrmOldPermParameters
- UrmOldPermRecLists
- UrmOldPermSave
- UrmOldPermSelections
- UrmOldPermTextMix
- UrmOldPermTransfers
- UrmPermConfig
- UrmPermCreate
- UrmPermDelete
- UrmPermDeleteOwner
- UrmPermElmGroupDelete
- UrmPermElmGroupInsert
- UrmPermElmGroupRead
- UrmPermExecute
- UrmPermMemberDelete
- UrmPermMemberInsert
- UrmPermModify
- UrmPermModifyOwner
- UrmPermRead
- UrmPermUser
- UrmPrev
- UrmStrict
- UrmTypeElmBlob
- UrmTypeElmCustom
- UrmTypeElmDialog
- UrmTypeElmElmGroup
- UrmTypeElmGroup
- UrmTypeElmMenu
- UrmTypeElmMetaPicture
- UrmTypeElmPicture
- UrmTypeElmPrintDocRecord
- UrmTypeElmPrintDocument
- UrmTypeElmPrintForm
- UrmTypeElmPrintFormList
- UrmTypeElmProcedure
- UrmTypeElmTable
- UrmTypeElmTheme
- UrmTypeElmUser



- UrmTypeElmUserGroup
- UrmTypeMember
- UrmTypePerm
- UrmTypeProperty
- UrmTypeSysProperty
- UrmTypeUser
- UrmTypeUserGroup
  
- UserAddress
- UserCurrent
- UserGroup
- UserJobID
- UserLastReq
- UserLastReqDate
- UserLastReqTime
- UserLocked
- UserLogin
- UserLoginDate
- UserLoginTime
- UserName
- UserNetAccount
- UserNextID
- UserNumber
- UserPlatform
- UserProtocol
- UserSysAccount
- UserSysName
- UserSysNameIP

## Kontakt

obj -> UrmCreate(int1, alpha2[, int3]) :



int

Objekt der Benutzerverwaltung erzeugen

obj Deskriptor des Eltern-Objekts oder 0

Typ des Objekts	
<u>UrmTypeUser</u>	Benutzer
<u>UrmTypeUserGroup</u>	Benutzergruppe
<u>UrmTypeElmGroup</u>	Elementgruppe
<u>UrmTypeProperty</u>	Benutzerdefinierte
int1	Eigenschaft
	Mitglied oder
	Benutzergruppe
	Element einer
<u>UrmTypeMember</u>	
<u>UrmTypeElm...</u>	Elementgruppe

alpha2 Name des Objekts

Typ der Eigenschaft (optional - nur  
wenn int1 = UrmTypeProperty)

<u>TypeAlpha</u>	Alphanumerisch
<u>TypeBigInt</u>	Ganzzahlig (64 Bit)
<u>TypeDate</u>	Datum
<u>TypeDecimal</u>	Dezimal
<u>TypeFloat</u>	Gleitkomma
<u>TypeInt</u>	Ganzzahlig (32 Bit)
<u>TypeLogic</u>	Logisch
<u>TypeTime</u>	Zeit

Resultat int Fehlerwert



Siehe Verwandte Befehle, Benutzerpflege,

UrmDelete()

Mit dieser Anweisung wird ein Objekt der Benutzerverwaltung erzeugt. Abhängig von dem in (int1) übergebenen Typ werden unterschiedliche Objekte erzeugt. Namen für Benutzer, Benutzergruppen und Elementgruppen dürfen nicht mit einem Unterstrich (\_) beginnen, keine Steuerzeichen (ASCII-Wert < 32) und keines der folgenden Zeichen beinhalten: ! \* ? : ; / ' " \. Folgende Objekte können erzeugt werden:

- Benutzer

Soll ein neuer Datenbank-Benutzer erzeugt werden, muss in (obj) 0 und in (int1)

UrmTypeUser übergeben werden. Als (alpha2) wird der Name des neuen Benutzers angegeben. Der Name des Benutzers darf maximal 20 Zeichen lang sein.

- Benutzergruppe

Soll eine neue Benutzergruppe angelegt werden, muss in (obj) 0 und in (int1)

UrmTypeUserGroup angegeben werden. Der Name der Benutzergruppe wird in (alpha2) übergeben. Der Namen der Benutzergruppe darf maximal 20 Zeichen lang sein.

- Elementgruppe

## Kontakt

Soll eine neue Elementgruppe angelegt werden, muss in (obj) 0 und in (int1)

UrmTypeElmGroup angegeben werden. Der Name der Elementgruppe wird in (alpha2) angegeben. Der Name der Elementgruppe darf maximal 40 Zeichen lang sein.

- **Eigenschaft**

Soll eine neue Eigenschaft definiert werden, muss das Objekt, dass die neue Eigenschaft bekommen soll in (obj) übergeben werden. Das entsprechende Objekt muss zuvor mit UrmOpen() geöffnet werden. In (int1) wird UrmTypeProperty und in (alpha2) der Name der Eigenschaft übergeben. Der Name darf nicht länger als 40 Zeichen sein. Der Typ der Eigenschaft muss in (int3) übergeben werden. Der Wert der Eigenschaft kann anschließend mit der Anweisung UrmPropSet() gesetzt werden.

- **Mitgliedschaft**

Wird in (int1) der Typ UrmTypeMember übergeben, kann damit ein Benutzer zu einer Benutzergruppe zugeordnet werden. Wird in (obj) eine Benutzergruppe übergeben, muss in (alpha2) der Name eines Benutzers angegeben werden. Wird in (obj) ein Benutzer übergeben, muss in (alpha2) der Name einer Benutzergruppe angegeben werden. Der in (obj) übergebene Deskriptor muss zuvor mit UrmOpen() geöffnet werden.

- **Eintrag in einer Elementgruppe**

Soll ein neuer Eintrag in eine Elementgruppe vorgenommen werden, muss in (obj) der Deskriptor der Elementgruppe übergeben werden. Die Elementgruppe muss zuvor mit der Anweisung UrmOpen() geöffnet worden sein. In (int1) wird der Typ des Elements mit einer UrmTypeElm...-Konstante angegeben. Der Name des Objekts wird in (alpha2) übergeben.



Zum Typ UrmTypeElmBlob können momentan keine Datenbankobjekte hinzugefügt werden.



Die Anzahl der einzelnen Objekte, Eigenschaften, Mitgliedschaften usw. sind beschränkt. Die Limitationen sind im Abschnitt Limitationen des Benutzersystems erläutert.

Änderungen in Bezug auf Mitgliedschaften von Benutzern in Benutzergruppen und Elementen in Elementgruppen wirken sich sofort auf das Benutzersystem aus. Wird also ein Benutzer einer Benutzergruppe zugeordnet, stehen sofort die neuen Berechtigungen zur Verfügung. Bei Änderungen von Berechtigungen wird zwischen dem neuen System, das vom CONZEPT 16-Server verwaltet wird und dem alten System, das vom Client verwaltet wird, unterschieden. Änderungen im neuen System wirken sich sofort aus, während Änderungen im alten System sich erst nach einer Neuanmeldung des Benutzers auswirken.

Nachdem das Objekt erzeugt wurde, kann es mit der Anweisung UrmOpen() geöffnet und bearbeitet werden.

Als Rückgabewert wird eine Fehlerkonstante zurückgegeben. Sie kann mit folgenden Konstanten verglichen werden:

## Kontakt

<u>ErrOk</u>	Kein Fehler aufgetreten.
<u>ErrExists</u>	Das Objekt ist bereits vorhanden.
<u>ErrNameInvalid</u>	In (alpha2) wurde ein falscher Name angegeben.
<u>ErrRights</u>	Der Benutzer verfügt nicht über ausreichende Rechte.
<u>ErrUrmParentNotFound</u>	Das in (obj) übergebene Eltern-Objekt wurde nicht gefunden.
<u>ErrUrmObjectNotFound</u>	Das in (alpha2) angegebene Objekt existiert nicht.
<u>ErrType</u>	In (alpha2) oder (int3) ist ein falscher Typ angegeben worden.
<u>ErrLimitExceeded</u>	Die <u>Limitationen</u> wurden überschritten.

### Beispiel

```
// Benutzer erzeugentErg # UrmCreate(0, _UrmTypeUser, 'Sales1');tHdlUser # UrmOpen(_UrmTypeUser,
```

### Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist nicht gültig.
<u>ErrValueInvalid</u>	In (int1) ist ein unbekannter Typ übergeben worden.
<u>ErrStringOverflow</u>	Der Objektname in (alpha2) ist zu lang.

## Kontakt

UrmOpen(int1, int2,

alpha3) : handle

Benutzerobjekt öffnen



Typ

int1      UrmTypeUser      Benutzer  
            UrmTypeUserGroup Benutzergruppe  
            UrmTypeElmGroup Elementgruppe

Sperrmodus

int2      0      nicht sperren  
            UrmLock      Sperre einrichten  
            UrmStrict Strikte

Rechtekontrolle

alpha3      Objektname

Resultat handle      Deskriptor oder  
                         Fehlerwert



Siehe      Verwandte Befehle,  
            Benutzerpflege, UrmClose()

Mit dieser Anweisung kann ein Objekt aus der Benutzerverwaltung geöffnet werden. Der Typ des zu öffnenden Objekts muss in (int1) übergeben werden. Folgende Objekte können geöffnet werden:

UrmTypeUser      Benutzer  
UrmTypeUserGroup Benutzergruppe  
UrmTypeElmGroup Elementgruppe

Über den Parameter (int2) kann eine Sperre eingerichtet werden. Im Gegensatz zu Datensatzsperrern kann das Objekt auch ohne eine Sperre verändert werden. Das Löschen eines gesperrten Objekts ist allerdings nicht möglich. Das Setzen der Sperre kann zur Synchronisation mehrerer Clients dienen. Mit UrmLock wird eine benutzerspezifische Sperre eingerichtet. Versucht der gleiche Benutzer noch einmal das Objekt zu sperren erfolgt keine Fehlermeldung.

Beim Öffnen eines Objekts kann ebenfalls eine strikte Rechtekontrolle aktiviert werden. Ein Benutzer hat normalerweise das Recht, eigene Eigenschaften zu lesen und zu verändern, sowie Eigenschaften seiner Benutzergruppen zu lesen. Durch die Angabe von UrmStrict in (int2) werden diese Sonderrechte nicht berücksichtigt.

In (alpha3) wird der Name des Objekts angegeben. Der Name kann zuvor mit UrmRead() ermittelt werden.

Das Resultat der Anweisung ist entweder der Deskriptor des Objekts, oder einer der folgenden negativen Fehlerwerte:

ErrLocked      Objekt ist von einem anderen Benutzer gesperrt.  
ErrUrmObjectNotFound Das in (alpha3) angegebene Objekt kann nicht gefunden werden.

### Beispiel

```
tHdl # UrmOpen(_UrmTypeUser, _UrmLock, 'SUPERUSER');if (tHdl < 0){ // Fehlerbehandlung ...}...t
```

## Kontakt

Mögliche Laufzeitfehler:

ErrValueInvalid In (int1) wurde ein ungültiger Typ übergeben.

ErrStringOverflow Der Objektname in (alpha3) ist zu lang.

## Kontakt

obj -> UrmClose()

Benutzerobjekt schließen

Deskriptor des

obj Objekts der

Benutzerpflege

Verwandte

Siehe Befehle,

Benutzerpflege,

UrmOpen()








Mit dieser Anweisung wird ein geöffnetes Objekt der Benutzerverwaltung wieder geschlossen. In (obj) wird der Deskriptor übergeben, der von UrmOpen() zurückgegeben wurde.

Die beim Öffnen des Objekts angegebene Sperre wird beim Schließen des Objekts mit entfernt.

### Beispiel

```
// Benutzerobjekt lesentHdlUsr # UrmOpen(_UrmTypeUser, _UrmLock, 'Sales');...// Benutzerobjekt sc
```

## Kontakt

obj -> UrmDelete(int1, alpha2) : int        Objekt der Benutzerverwaltung löschen

obj      Deskriptor des Eltern-Objekts oder 0

Typ des Objekts

UrmTypeUser

    Benutzer löschen

UrmTypeUserGroup

    Benutzergruppe

    löschen

UrmTypeElmGroup

    Elementgruppe

    löschen

int1

UrmTypeProperty

    Benutzerdefinierte  
    Eigenschaft

    löschen

UrmTypeMember

    Mitglied oder  
    Benutzergruppe

    entfernen

UrmTypeElm

    Element löschen

alpha2    Name des Objekts

Resultat int    Fehlerwert



Siehe

Verwandte Befehle, Benutzerpflege,

UrmCreate()

Mit dieser Anweisung wird ein Objekt der Benutzerverwaltung gelöscht. Folgende Objekte können gelöscht werden:

- Benutzer

Soll ein vorhandener Datenbank-Benutzer gelöscht werden, muss in (obj) 0 und in (int1)

UrmTypeUser übergeben werden. In (alpha2) wird der Name des Benutzers angegeben. Der Benutzer wird automatisch aus allen Benutzergruppen entfernt.

- Benutzergruppe

Soll eine Benutzergruppe gelöscht werden, muss in (obj) 0 und in (int1)

UrmTypeUserGroup angegeben werden. Der Name der Benutzergruppe wird in (alpha2) übergeben. Die Liste der Benutzergruppen bei den einzelnen Benutzern wird automatisch aktualisiert.

- Elementgruppe

Soll eine Elementgruppe gelöscht werden, muss in (obj) 0 und in (int1)

UrmTypeElmGroup angegeben werden. Der Name der Elementgruppe wird in (alpha2) angegeben. Die Elementgruppe wird automatisch bei allen Benutzergruppen entfernt.

- Eigenschaft

Soll eine Eigenschaft gelöscht werden, muss das Objekt, dem die Eigenschaft gehört in (obj) übergeben werden. Das entsprechende Objekt muss zuvor mit UrmOpen() gelesen werden.

In (int1) wird UrmTypeProperty und in (alpha2) der Name der Eigenschaft übergeben.

Eigenschaften, die vom System zur Verfügung gestellt werden, können nicht gelöscht werden.

- Mitgliedschaft



## Kontakt

Wird in (int1) der Typ UrmTypeMember übergeben, kann damit ein Benutzer aus einer Benutzergruppe entfernt werden. Wird in (obj) eine Benutzergruppe übergeben, muss in (alpha2) der Name des Benutzers angegeben werden. Wird in (obj) ein Benutzer übergeben, muss in (alpha2) der Name der Benutzergruppe angegeben werden. Der in (obj) übergebene Deskriptor muss zuvor mit UrmOpen() geöffnet werden.

- Eintrag in einer Elementgruppe

Soll ein Eintrag aus einer Elementgruppe entfernt werden, muss in (obj) der Deskriptor der Elementgruppe übergeben werden. Die Elementgruppe muss zuvor mit der Anweisung UrmOpen() gelesen worden sein. In (int1) wird der Typ des Eintrags mit einer UrmTypeElm...-Konstante angegeben. Der Name des Eintrags wird in (alpha2) übergeben.

Über den Rückgabewert kann der Erfolg des Befehls überprüft werden:

<u>ErrOk</u>	Kein Fehler aufgetreten.
<u>ErrRights</u>	Der Benutzer verfügt nicht über ausreichende Rechte.
<u>ErrType</u>	In (alpha2) ist ein falscher Element-Typ angegeben worden.
<u>ErrUrmParentNotFound</u>	Das in (obj) übergebene Eltern-Objekt wurde nicht gefunden.
<u>ErrUrmObjectNotFound</u>	Das in (alpha2) übergebene Objekt wurde nicht gefunden.
<u>ErrUnerasable</u>	Das Objekt kann nicht gelöscht werden.
<u>ErrLocked</u>	Das Objekt ist von einem anderen Benutzer gesperrt.
<u>ErrInUse</u>	Bei (int1 = <u>_UrmTypeUser</u> ) wurde ein Benutzer angegeben, der zurzeit angemeldet ist.

### Beispiel

```
// Benutzer löschtErg # UrmDelete(0, _UrmTypeUser, 'Sales1');// Benutzer aus einer Benutzergrup
```

### Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig.
<u>ErrValueInvalid</u>	Der in (obj) übergeben Deskriptor hat den falschen Typ.
<u>ErrStringOverflow</u>	Der Objektname in (alpha2) ist zu lang.

## Kontakt

obj -> UrmPermGet(alpha1, int2, bigint3) : logic

Ermitteln, ob bestimmte Rechte vorhanden sind

obj Deskriptor der

Benutzergruppe oder 0

alpha1 Name der Elementgruppe

Modus

UrmAllow Recht

int2 erlauben

UrmDeny Recht

verbieten

bigint3 Berechtigungsflags

Resultat logic Rechte vorhanden

Verwandte Befehle,  
Benutzerpflege,

Siehe

UrmPermGetRaw(),

UrmPermSet()

Mit dieser Anweisung kann überprüft werden, ob der eigene Benutzer oder eine Benutzergruppe über bestimmte Berechtigungen auf einer Elementgruppe verfügt oder sie entzogen bekommen hat.

- Rechte einer Benutzergruppe ermitteln

Der Deskriptor der Benutzergruppe wird in (obj) übergeben. In (alpha1) wird der Name der Elementgruppe angegeben, deren Rechte ermittelt werden sollen.

- Die eigenen Rechte ermitteln

Um die eigenen Berechtigungen zu ermitteln, wird als Deskriptor in (obj) 0 übergeben. Dabei werden dann die effektiven Rechte überprüft, die sich aus der Kombination der Rechte aller Benutzergruppen zusammen setzt, der der Benutzer zugeordnet ist. Der Name der Elementgruppe wird in (alpha1) übergeben. Es können nur die erlaubten Rechte (int2 = UrmAllow) ermittelt werden.

In (bigint3) wird eine beliebige Kombination von Berechtigungsflags und in (int2) der Abfragemodus übergeben. Die in (bigint3) übergebene Kombination aus Berechtigungsflags kann mit folgenden Konstanten zusammengestellt werden:

<u>UrmOldPermAccess</u>	Dateiberechtigung zum Zugriff auf Datensätze
<u>UrmOldPermDelete</u>	Dateiberechtigung zum Löschen von Datensätzen
<u>UrmOldPermEntry</u>	Dateiberechtigung zur Eingabe von Datensätzen
<u>UrmOldPermExecLists</u>	Dateiberechtigung Ausführen von Listen
<u>UrmOldPermExecSelections</u>	Dateiberechtigung Ausführen von Selektionen
<u>UrmOldPermExecTransfers</u>	Dateiberechtigung zur Ausführung von Transfers
<u>UrmOldPermLink</u>	Dateiberechtigung zum Zugriff auf Verknüpfungen
<u>UrmOldPermListFormats</u>	Dateiberechtigung zur Änderung von Listenformaten
<u>UrmOldPermModify</u>	Dateiberechtigung zum Ändern von Datensätzen
<u>UrmOldPermParameters</u>	Dateiberechtigung zur Änderung von Dateiparametern

## Kontakt

<u>UrmOldPermRecLists</u>	Dateiberechtigung zum Ändern von Zugriffslisten
<u>UrmOldPermSave</u>	Dateiberechtigung zum Speichern von Datensätzen
<u>UrmOldPermSelections</u>	Dateiberechtigung zum Ändern von Selektionen
<u>UrmOldPermTextMix</u>	Dateiberechtigung zum Text und Daten mischen
<u>UrmOldPermTransfers</u>	Dateiberechtigung zum Ändern von Transfers
<u>UrmPermRead</u>	Berechtigung zum Lesen
<u>UrmPermCreate</u>	Berechtigung zum Erzeugen
<u>UrmPermModify</u>	Berechtigung zum Ändern
<u>UrmPermModifyOwner</u>	Berechtigung zum Ändern bei Objektbesitz
<u>UrmPermDelete</u>	Berechtigung zum Löschen
<u>UrmPermDeleteOwner</u>	Berechtigung zum Löschen bei Objektbesitz
<u>UrmPermExecute</u>	Berechtigung zum Ausführen
<u>UrmPermConfig</u>	Berechtigung zur Änderung der Tabellenstruktur
<u>UrmIdePermRead</u>	Berechtigung zum Lesen (Entwicklungsumgebung)
<u>UrmIdePermCreate</u>	Berechtigung zum Erzeugen (Entwicklungsumgebung)
<u>UrmIdePermModify</u>	Berechtigung zum Ändern von (Entwicklungsumgebung)
<u>UrmIdePermDelete</u>	Berechtigung zum Löschen von (Entwicklungsumgebung)
<u>UrmPermElmGroupRead</u>	Berechtigung Elementgruppen lesen
<u>UrmPermElmGroupInsert</u>	Berechtigung Element zur Gruppe hinzufügen
<u>UrmPermElmGroupDelete</u>	Berechtigung Element aus Gruppe löschen
<u>UrmPermMemberInsert</u>	Berechtigung Benutzer zu Benutzergruppe hinzufügen
<u>UrmPermMemberDelete</u>	Berechtigung Benutzer aus Benutzergruppe entfernen

Die Berechtigungen der alten Benutzerverwaltung können mit Hilfe von \_UrmOldPerm...-Konstanten ermittelt werden. Die Konstanten sind im Abschnitt Konvertierung des alten Benutzersystems zusammengefasst.

Der Programmierer kann eigene Rechte definieren. Dazu stehen ihm die

UrmPermUser-Konstanten zur Verfügung.

Je nach übergebenem Modus (int2) kann überprüft werden, ob eine Benutzergruppe ein oder mehrere Rechte zugesprochen oder entzogen bekommen hat. Mit dem Aufruf `UrmPermGet(..., _UrmAllow, ...)` werden die zugesicherten Rechte überprüft, mit dem Aufruf `UrmPermGet(..., _UrmDeny, ...)` die entzogenen Rechte.

Werden mehrere Rechte gleichzeitig überprüft, gibt die Anweisung nur dann true zurück, wenn mindestens die angegebenen Rechte zugesprochen oder entzogen wurden. Sind bei einer Elementgruppe zum Beispiel die Rechte UrmPermRead, UrmPermCreate und UrmPermModify erlaubt, wird bei der Überprüfung mit `UrmPermGet(..., _UrmAllow, _UrmCreate | _UrmModify)` true zurückgegeben. Bei der

## Kontakt

Überprüfung mit `UrmPermGet(..., _UrmAllow, _UrmCreate | _UrmExecute)` wird allerdings false zurückgegeben, weil das Ausführungsrecht nicht zugesichert ist.

### Beispiel

```
// Überprüfung der eigenen Rechteif (UrmPermGet(0, 'Customer', _UrmAllow, _UrmPermRead)){ // Anz
```

### Mögliche Laufzeitfehler:

- \_ErrHdlInvalid**      Der in (obj) übergebene Deskriptor ist ungültig.
- \_ErrStringOverflow** Es wurde ein zu langer Elementname angegeben.
- \_ErrValueInvalid**      Es wurde ein unbekannter Modus in (int2) übergeben.

obj ->

UrmPermGetRaw(alpha1, )

int2) : bigint

Alle Rechte ermitteln

obj Deskriptor der

Benutzergruppe oder 0

alpha1 Name der Elementgruppe

Modus

int2 UrmAllow Recht erlauben

UrmDeny Recht verbieten

Resultat bigint Berechtigungsflags 

Verwandte Befehle,

Benutzerpflege,

Siehe

UrmPermGet(),

UrmPermSet()

Mit dieser Anweisung kann ermittelt werden, welche Berechtigungen auf einer Elementgruppe der eigene Benutzer oder eine Benutzergruppe zugesichert oder entzogen bekommen hat.

- Rechte einer Benutzergruppe ermitteln

Der Deskriptor der Benutzergruppe wird in (obj) übergeben. In (alpha1) wird der Name der Elementgruppe angegeben, deren Rechte ermittelt werden sollen.

- Die eigenen Rechte ermitteln

Um die eigenen Berechtigungen zu ermitteln, wird als Deskriptor in (obj) 0 übergeben. Dabei werden dann die effektiven Rechte überprüft, die sich aus der Kombination der Rechte aller Benutzergruppen zusammen setzt, der der Benutzer zugeordnet ist. Der Name der Elementgruppe wird in (alpha1) übergeben. Es können nur die erlaubten Rechte (int2 = UrmAllow) ermittelt werden.

In (int2) wird angegeben, ob die zugesicherten ( UrmAllow) oder die entzogenen ( UrmDeny) Rechte ermittelt werden sollen.

Das Resultat entspricht einem Bit-Muster, das an den Stellen auf 1 gesetzt ist, an denen das betreffende Recht auf den übergebenen Modus gesetzt ist. Zur Abfrage der Rechte stehen folgende Konstanten zur Verfügung:

<u>UrmOldPermAccess</u>	Dateiberechtigung zum Zugriff auf Datensätze
<u>UrmOldPermDelete</u>	Dateiberechtigung zum Löschen von Datensätzen
<u>UrmOldPermEntry</u>	Dateiberechtigung zur Eingabe von Datensätzen
<u>UrmOldPermExecLists</u>	Dateiberechtigung Ausführen von Listen
<u>UrmOldPermExecSelections</u>	Dateiberechtigung Ausführen von Selektionen
<u>UrmOldPermExecTransfers</u>	Dateiberechtigung zur Ausführung von Transfers
<u>UrmOldPermLink</u>	Dateiberechtigung zum Zugriff auf Verknüpfungen
<u>UrmOldPermListFormats</u>	Dateiberechtigung zur Änderung von Listenformaten

## Kontakt

<u>_UrmOldPermModify</u>	Dateiberechtigung zum Ändern von Datensätzen
<u>_UrmOldPermParameters</u>	Dateiberechtigung zur Änderung von Dateiparametern
<u>_UrmOldPermRecLists</u>	Dateiberechtigung zum Ändern von Zugriffslisten
<u>_UrmOldPermSave</u>	Dateiberechtigung zum Speichern von Datensätzen
<u>_UrmOldPermSelections</u>	Dateiberechtigung zum Ändern von Selektionen
<u>_UrmOldPermTextMix</u>	Dateiberechtigung zum Text und Daten mischen
<u>_UrmOldPermTransfers</u>	Dateiberechtigung zum Ändern von Transfers
<u>_UrmPermRead</u>	Berechtigung zum Lesen
<u>_UrmPermCreate</u>	Berechtigung zum Erzeugen
<u>_UrmPermModify</u>	Berechtigung zum Ändern
<u>_UrmPermModifyOwner</u>	Berechtigung zum Ändern bei Objektbesitz
<u>_UrmPermDelete</u>	Berechtigung zum Löschen
<u>_UrmPermDeleteOwner</u>	Berechtigung zum Löschen bei Objektbesitz
<u>_UrmPermExecute</u>	Berechtigung zum Ausführen
<u>_UrmPermConfig</u>	Berechtigung zur Änderung der Tabellenstruktur
<u>_UrmIdePermRead</u>	Berechtigung zum Lesen (Entwicklungsumgebung)
<u>_UrmIdePermCreate</u>	Berechtigung zum Erzeugen (Entwicklungsumgebung)
<u>_UrmIdePermModify</u>	Berechtigung zum Ändern von (Entwicklungsumgebung)
<u>_UrmIdePermDelete</u>	Berechtigung zum Löschen von (Entwicklungsumgebung)
<u>_UrmPermElmGroupRead</u>	Berechtigung Elementgruppen lesen
<u>_UrmPermElmGroupInsert</u>	Berechtigung Element zur Gruppe hinzufügen
<u>_UrmPermElmGroupDelete</u>	Berechtigung Element aus Gruppe löschen
<u>_UrmPermMemberInsert</u>	Berechtigung Benutzer zu Benutzergruppe hinzufügen
<u>_UrmPermMemberDelete</u>	Berechtigung Benutzer aus Benutzergruppe entfernen

Die Berechtigungen der alten Benutzerverwaltung können mit Hilfe von \_UrmOldPerm...-Konstanten ermittelt werden. Die Konstanten sind im Abschnitt Konvertierung des alten Benutzersystems zusammengefasst.

Sind eigene Rechte definiert, können zusätzlich die \_UrmPermUser...-Konstanten verwendet werden.

### Beispiel

```
// Überprüfung der eigenen Rechteif (UrmPermGetRaw(0, 'Customer', _UrmAllow) & _UrmPermRead > 0\b
```

### Mögliche Laufzeitfehler:

<u>_ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig.
-----------------------	--

## Kontakt

ErrValueInvalid Es wurde ein unbekannter Modus in (int2) übergeben.

ErrStringOverflow Es wurde ein zu langer Objektname angegeben.

## Kontakt

**UrmPermElementGet(int1,  
alpha2, bigint3) : logic**  
Element-Rechte ermitteln



int1                    Elementtyp  
                  ( UrmTypeElm...) alpha2

Name des Elements bigint3  
Berechtigungsflags Resultat logic  
Rechte vorhanden



Verwandte Befehle,  
Siehe            Benutzerpflege,  
                  UrmPermElementGetRaw()

Mit dieser Anweisung können die eigenen Rechte überprüft werden. In (int1) wird der Elementtyp und in (alpha2) der Name des Elementes (siehe Benutzerpflege) übergeben. In (bigint3) wird eine beliebige Kombination von Berechtigungsflags angegeben. Die übergebene Kombination aus Berechtigungsflags kann mit folgenden Konstanten zusammengestellt werden:

<u>_UrmPermRead</u>	Berechtigung zum Lesen
<u>_UrmPermCreate</u>	Berechtigung zum Erzeugen
<u>_UrmPermModify</u>	Berechtigung zum Ändern
<u>_UrmPermModifyOwner</u>	Berechtigung zum Ändern bei Objektbesitz
<u>_UrmPermDelete</u>	Berechtigung zum Löschen
<u>_UrmPermDeleteOwner</u>	Berechtigung zum Löschen bei Objektbesitz
<u>_UrmPermExecute</u>	Berechtigung zum Ausführen
<u>_UrmPermConfig</u>	Berechtigung zur Änderung der Datenstruktur

<u>_UrmPermElmGroupRead</u>	Berechtigung Elementgruppen lesen
<u>_UrmPermElmGroupInsert</u>	Berechtigung Element zur Gruppe hinzufügen
<u>_UrmPermElmGroupDelete</u>	Berechtigung Element aus Gruppe löschen

<u>_UrmPermElmGroupRead</u>	Berechtigung Benutzer zu Benutzergruppe hinzufügen
<u>_UrmPermElmGroupInsert</u>	Berechtigung Benutzer aus Benutzergruppe entfernen

Der Programmierer kann eigene Rechte definieren. Dazu stehen ihm die

\_UrmPermUser-Konstanten zur Verfügung.

Ist der aktuelle Benutzer im Besitz aller übergebenen Rechte, gibt der Befehl true zurück. Ist mindestens eines der übergebenen Rechte nicht vorhanden, wird false zurück gegeben.

Ist das übergebene Element nicht vorhanden, wird false zurück gegeben.

### Beispiel

```
// Darf ich eine Prozedur ausführen?if (UrmPermElementGet(_UrmTypeElmProcedure, 'LibDbServices',
```

Mögliche Laufzeitfehler:



## Kontakt

**ErrStringOverflow** Es wurde ein zu langer (mehr als 80 Zeichen) Elementname angegeben.

**ErrValueInvalid** Es wurde eine unbekannte Berechtigung (bigint2) übergeben.

## Kontakt

UrmPermElementGetRaw(int1, )

alpha2) : bigint

Element-Rechte ermitteln

int1 Elementtyp

( UrmTypeElm...)

alpha2 Name des Elements

Resultat bigint Berechtigungsflags

Verwandte Befehle,

Siehe Benutzerpflege,

UrmPermElementGet()

Mit dieser Anweisung können die eigenen Rechte überprüft werden. In (int1) wird der Elementtyp und in (alpha2) der Name des Elementes (siehe Benutzerpflege) übergeben. Als Rückgabewert kommt ein bigint zurück, das alle Berechtigungsflags als Kombination der folgenden Konstanten enthält:

<u>_UrmPermRead</u>	Berechtigung zum Lesen
<u>_UrmPermCreate</u>	Berechtigung zum Erzeugen
<u>_UrmPermModify</u>	Berechtigung zum Ändern
<u>_UrmPermModifyOwner</u>	Berechtigung zum Ändern bei Objektbesitz
<u>_UrmPermDelete</u>	Berechtigung zum Löschen
<u>_UrmPermDeleteOwner</u>	Berechtigung zum Löschen bei Objektbesitz
<u>_UrmPermExecute</u>	Berechtigung zum Ausführen
<u>_UrmPermConfig</u>	Berechtigung zur Änderung der Datenstruktur

<u>_UrmPermElmGroupRead</u>	Berechtigung Elementgruppen lesen
<u>_UrmPermElmGroupInsert</u>	Berechtigung Element zur Gruppe hinzufügen
<u>_UrmPermElmGroupDelete</u>	Berechtigung Element aus Gruppe löschen

<u>_UrmPermElmGroupRead</u>	Berechtigung Benutzer zu Benutzergruppe hinzufügen
<u>_UrmPermElmGroupInsert</u>	Berechtigung Benutzer aus Benutzergruppe entfernen

Der Programmierer kann eigene Rechte definieren. Dazu stehen ihm die

\_UrmPermUser-Konstanten zur Verfügung.

Das Ergebnis kann mit den Konstanten verglichen werden und so festgestellt werden, welche Rechte vorhanden sind.

Ist das übergebene Element nicht vorhanden, wird 0 zurück gegeben.

Beispiel

```
// Darf ich eine Prozedur ausführen?if (UrmPermElementGetRaw(_UrmTypeElmProcedure, 'LibDbServices
```

Mögliche Laufzeitfehler:

ErrStringOverflow Es wurde ein zu langer (mehr als 80 Zeichen) Elementname angegeben.

obj ->

**UrmPermSet(alpha1,  
int2, bigint3) : int**

**Bestimmte Rechte setzen**

obj      Deskriptor der

Benutzergruppe

alpha1    Name der  
Elementgruppe  
Modus

UrmAllow Recht

int2      erlauben

UrmDeny Recht

verbieten

bigint3    Berechtigungsflags

Resultat int Fehlerwert

Verwandte Befehle,  
Benutzerpflege,

Siehe

UrmPermGet(),  
UrmPermGetRaw()

Mit dieser Anweisung können Rechte von Benutzergruppen auf Elementgruppen gesetzt oder entzogen werden. Der Deskriptor der Benutzergruppe wird in (obj) übergeben. Der Name der Elementgruppe wird in (alpha1) übergeben.

In (bigint3) wird eine beliebige Kombination von Berechtigungsflags und in (int2) der Modus übergeben. Ein Recht kann sowohl mit UrmAllow erlaubt, als auch mit UrmDeny verboten werden.



Sind für ein Recht beide Modi gesetzt, wird das Recht verboten.

Dies kann verwendet werden, um einem Benutzer, der in zwei Benutzergruppen ist explizit Rechte der ersten Benutzergruppe zu verbieten, die er nicht haben soll.

Die in (bigint3) übergebene Kombination aus Berechtigungsflags kann mit folgenden Konstanten zusammengestellt werden:

<u>UrmOldPermAccess</u>	Dateiberechtigung zum Zugriff auf Datensätze
<u>UrmOldPermDelete</u>	Dateiberechtigung zum Löschen von Datensätzen
<u>UrmOldPermEntry</u>	Dateiberechtigung zur Eingabe von Datensätzen
<u>UrmOldPermExecLists</u>	Dateiberechtigung Ausführen von Listen
<u>UrmOldPermExecSelections</u>	Dateiberechtigung Ausführen von Selektionen
<u>UrmOldPermExecTransfers</u>	Dateiberechtigung zur Ausführung von Transfers
<u>UrmOldPermLink</u>	Dateiberechtigung zum Zugriff auf Verknüpfungen
<u>UrmOldPermListFormats</u>	Dateiberechtigung zur Änderung von Listenformaten
<u>UrmOldPermModify</u>	Dateiberechtigung zum Ändern von Datensätzen
<u>UrmOldPermParameters</u>	Dateiberechtigung zur Änderung von Dateiparametern
<u>UrmOldPermRecLists</u>	Dateiberechtigung zum Ändern von Zugriffslisten
<u>UrmOldPermSave</u>	Dateiberechtigung zum Speichern von Datensätzen

## Kontakt

<u>_UrmOldPermSelections</u>	Dateiberechtigung zum Ändern von Selektionen
<u>_UrmOldPermTextMix</u>	Dateiberechtigung zum Text und Daten mischen
<u>_UrmOldPermTransfers</u>	Dateiberechtigung zum Ändern von Transfers
<u>_UrmPermRead</u>	Berechtigung zum Lesen
<u>_UrmPermCreate</u>	Berechtigung zum Erzeugen
<u>_UrmPermModify</u>	Berechtigung zum Ändern
<u>_UrmPermModifyOwner</u>	Berechtigung zum Ändern bei Objektbesitz
<u>_UrmPermDelete</u>	Berechtigung zum Löschen
<u>_UrmPermDeleteOwner</u>	Berechtigung zum Löschen bei Objektbesitz
<u>_UrmPermExecute</u>	Berechtigung zum Ausführen
<u>_UrmPermConfig</u>	Berechtigung zur Änderung der Tabellenstruktur
<u>_UrmIdePermRead</u>	Berechtigung zum Lesen (Entwicklungsumgebung)
<u>_UrmIdePermCreate</u>	Berechtigung zum Erzeugen (Entwicklungsumgebung)
<u>_UrmIdePermModify</u>	Berechtigung zum Ändern von (Entwicklungsumgebung)
<u>_UrmIdePermDelete</u>	Berechtigung zum Löschen von (Entwicklungsumgebung)
<u>_UrmPermElmGroupRead</u>	Berechtigung Elementgruppen lesen
<u>_UrmPermElmGroupInsert</u>	Berechtigung Element zur Gruppe hinzufügen
<u>_UrmPermElmGroupDelete</u>	Berechtigung Element aus Gruppe löschen

<u>_UrmPermMemberInsert</u>	Berechtigung Benutzer zu Benutzergruppe hinzufügen
<u>_UrmPermMemberDelete</u>	Berechtigung Benutzer aus Benutzergruppe entfernen

Die Berechtigungen der alten Benutzerverwaltung können mit Hilfe von

\_UrmOldPerm...-Konstanten gesetzt werden. Die Konstanten sind im Abschnitt Konvertierung des alten Benutzersystems zusammengefasst.

Der Programmierer kann eigene Rechte definieren. Dazu stehen ihm die

\_UrmPermUser...-Konstanten zur Verfügung.

Für jeden Modus muss die Anweisung separat aufgerufen werden. Sollen also bestimmte Rechte gesetzt und andere Entzogen werden, muss die Anweisung zwei mal aufgerufen werden.

Bei Änderungen von Berechtigungen wird zwischen dem neuen System, das vom CONZEPT 16-Server verwaltet wird und dem alten System, das vom Client verwaltet wird, unterschieden. Änderungen im neuen System wirken sich sofort aus, während Änderungen im alten System sich erst nach einer Neuanmeldung des Benutzers auswirken.

Der Rückgabewert des Befehls kann mit folgenden Konstanten verglichen werden:

## Kontakt

<u>ErrOk</u>	kein Fehler aufgetreten
<u>ErrRights</u>	Berechtigung nicht ausreichend
<u>ErrUrmObjectNotFound</u>	Objekt (obj) nicht mehr vorhanden
<u>ErrUnchangeable</u>	Die Berechtigungen dürfen nicht verändert werden

### Beispiel

```
// Rechte der Gruppe neu definierenHdlUserGrp->UrmPermSet('Customer', _UrmAllow, _UrmPermRead |
```

### Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig.
<u>ErrStringOverflow</u>	Es wurde ein zu langer Objektname angegeben.
<u>ErrValueInvalid</u>	Es wurde ein unbekannter Modus in (int2) übergeben.

## Kontakt

obj -> UrmPermLevelGet(alpha1) : int  Benutzerlevel aus dem alten Benutzersystem ermitteln

obj      Deskriptor der Benutzergruppe oder 0

alpha1    Name der Elementgruppe

Resultat int Benutzerlevel

Verwandte

Befehle,

Benutzerpflege,

Siehe UrmPermLevelSet(),  
Konvertierung des  
alten  
Benutzersystems

Diese Funktion ermittelt den Berechtigungslevel einer Elementgruppe. Der Befehl wird benötigt, um die Abwärtskompatibilität zum alten Benutzersystem zu gewährleisten.

Mit dieser Anweisung kann ermittelt werden, über welchen Berechtigungslevel eine Benutzergruppe in Bezug auf eine Elementgruppe verfügt. Der Deskriptor der Benutzergruppe wird in (obj) übergeben. Wird als Deskriptor 0 übergeben, wird der eigene Benutzerlevel ermittelt. Der Name der Elementgruppe wird in (alpha1) übergeben.

Das Resultat entspricht dem Berechtigungslevel des alten Benutzersystems. Der Level liegt zwischen 0 und 250.

Mögliche Laufzeitfehler:

ErrHdlInvalid

Der in (obj) übergebene Deskriptor ist ungültig.

ErrStringOverflow

Es wurde ein zu langer Objektname angegeben.

## Kontakt

obj -> UrmPermLevelSet(alpha1, int2) : int        Benutzerlevel aus dem alten Benutzersystem ermitteln

obj            Deskriptor der  
              Benutzergruppe

alpha1    Name der  
           Elementgruppe  
           Neuer

int2            Benutzerlevel

Resultat int Fehlerwert 

Verwandte  
              Befehle,  
              Benutzerpflege,  
Siehe       UrmPermLevelGet(),  
              Konvertierung des  
              alten  
              Benutzersystems

Diese Funktion setzt den Berechtigungslevel einer Elementgruppe. Der Befehl wird benötigt, um die Abwärtskompatibilität zum alten Benutzersystem zu gewährleisten.

Mit dieser Anweisung kann der Berechtigungslevel einer Benutzergruppe in Bezug auf eine Elementgruppe gesetzt werden. Der Deskriptor der Benutzergruppe wird in (obj) übergeben. Der Name der Elementgruppe wird in (alpha1) übergeben. In (int2) wird der neue Berechtigungslevel angegeben. Der Level muss zwischen 0 und 250 (einschließlich) liegen.

Der Erfolg des Befehls kann über seinen Rückgabewert kontrolliert werden.

<u>ErrOk</u>	Kein Fehler aufgetreten
<u>ErrRights</u>	Berechtigung nicht ausreichend
<u>ErrUrmObjectNotFound</u>	Das angegebene Objekt (obj) wurde nicht gefunden
<u>ErrUnchangeable</u>	Berechtigungslevel kann nicht geändert werden


Beispiel

```
// Recht setzentErg # tHdlUserGroup->UrmPermLevelSet('table:tblCstCustomer', 100)
```

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig.
<u>ErrStringOverflow</u>	Es wurde ein zu langer Objektname angegeben.

## Kontakt

obj -> UrmPropGet(alpha1, var2) : logic  Lesen einer  
Eigenschaft in der Benutzerverwaltung

obj            Deskriptor des  
              Objekts  
              Name oder  
alpha1       Konstante der  
              Eigenschaft  
var2          Variable  
Resultat logic Erfolg  
              Verwandte  
Siehe        Befehle,  
              UrmPropSet()

Dieser Befehl liest eine Eigenschaft eines Objekts der Benutzerverwaltung aus. Als (obj) wird der Deskriptor des Objekts angegeben. Der Deskriptor wird beim Öffnen des Objekts (UrmOpen()) zurück gegeben.

In (alpha1) wird der Name der Eigenschaft angegeben. Bei den vom System definierten Eigenschaften (Systemproperties) der Objekte kann auch eine entsprechende Konstante angegeben werden. Die Konstante wird aus \_UrmProp und dem Namen der Eigenschaft zusammengesetzt (siehe Eigenschaften von Objekten des Benutzersystems).

Der Wert der Eigenschaft wird in der Variable (var2) gespeichert. Die Variable muss den gleichen Typ besitzen, wie die Eigenschaft. Der Typ der Eigenschaft kann mit Hilfe der Anweisung UrmPropType() ermittelt werden.

Der Erfolg der Anweisung kann über den Rückgabewert überprüft werden. Konnte der Wert der Eigenschaft ausgelesen werden, wird true zurück gegeben. Ist ein Fehler aufgetreten, gibt die Anweisung false zurück. In diesem Fall kann der Fehlerwert mit ErrGet() abgefragt werden. Ein Fehler führt zum Verlassen eines try-Blocks.

### Beispiel

```
if (!tHdlUser->UrmPropGet(_UrmPropName, tUsername)){ // Fehlerbehandlung}
```

### Mögliche Laufzeitfehler:


ErrHdlInvalid        Der angegebene Deskriptor in (obj) ist ungültig.

ErrNameInvalid      Der in (alpha1) angegebene Name ist nicht vorhanden.

ErrType              Die in (var2) angegebene Variable hat nicht den gleichen Typ wie die Eigenschaft.



## Kontakt

**obj** -> **UrmPropSet(alpha1, var2) : logic**  **Setzen einer**  
**Eigenschaft in der Benutzerverwaltung**  
**obj** Deskriptor des  
Objekts  
Name oder  
**alpha1** Konstante der  
Eigenschaft  
**var2** Wert  
Resultat **logic** Ergebnis  
Verwandte  
Siehe Befehle,  
UrmPropGet()

Dieser Befehl setzt eine Eigenschaft eines Objekts der Benutzerverwaltung. Als (**obj**) wird der Deskriptor des Objekts angegeben. Der Deskriptor wird beim Öffnen des Objekts (UrmOpen()) zurück gegeben.

In (**alpha1**) wird der Name der Eigenschaft angegeben. Bei den vordefinierten Eigenschaften (Systemproperties) der Objekte kann auch eine entsprechende Konstante angegeben werden. Die Konstante wird aus \_UrmProp und dem Namen der Eigenschaft zusammengesetzt (siehe Eigenschaften von Objekten des Benutzersystems)

Der neue Wert der Eigenschaft wird in (**var2**) übergeben. Der Wert muss den gleichen Typ besitzen, wie die Eigenschaft. Der Typ der Eigenschaft kann mit Hilfe der Anweisung UrmPropType() ermittelt werden.

Damit eine Eigenschaft gesetzt werden kann, müssen verschiedene Bedingungen erfüllt sein: Die Eigenschaft muss vorhanden sein, der Benutzer muss über ausreichende Rechte zum Schreiben der Eigenschaft verfügen und die Eigenschaft muss beschreibbar sein. Der Erfolg der Anweisung kann über den Rückgabewert überprüft werden. Konnte der Wert der Eigenschaft gesetzt werden, wird true zurück gegeben. Ist ein Fehler aufgetreten, gibt die Anweisung false zurück. In diesem Fall kann der Fehlerwert mit ErrGet() abgefragt werden. Ein Fehler führt zum Verlassen eines try-Blocks.

### Beispiel

```
if (!tHdlUser->UrmPropSet(_UrmPropActive, true)){ // Fehlerbehandlung} // benutzerdefinierte Eige
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebene Deskriptor in (**obj**) ist ungültig.  
ErrNameInvalid Die in (**alpha1**) angegebene Eigenschaft ist nicht vorhanden.  
ErrType Die Variable in (**var2**) hat einen anderen Typ als die Eigenschaft.

## Kontakt

**obj -> UrmPropType(alpha1) :**



**int**

**Typ einer Eigenschaft ermitteln**

Deskriptor eines Objekts

obj

der Benutzerverwaltung

**alpha1** Name oder Konstante der

Eigenschaft

**Resultat** int Typ der Eigenschaft  —

Verwandte Befehle,

Siehe

UrmCreate(), UrmOpen()

Mit dieser Anweisung kann der Typ einer Eigenschaft von einem Objekt der Benutzerverwaltung ermittelt werden. In (obj) wird der Deskriptor des Objekts angegeben, zu dem die Eigenschaft gehört. Der Deskriptor wird beim Öffnen (UrmOpen()) des Objekts zurück gegeben.

Bei den vordefinierten Eigenschaften der Objekte kann in (alpha1) eine entsprechende Konstante angegeben werden. Diese setzt sich aus \_UrmProp und dem Namen der Eigenschaft zusammen (siehe Eigenschaften von Objekten des Benutzersystems). Bei selbst definierten Eigenschaften wird hier der Name der Eigenschaft übergeben.

Der Rückgabewert kann mit folgenden Konstanten verglichen werden:











<u>_TypeAlpha</u>	Alphanumerisch
<u>_TypeBigInt</u>	Ganzzahlig (64 Bit)
<u>_TypeDate</u>	Datum
<u>_TypeDecimal</u>	Dezimal
<u>_TypeFloat</u>	Gleitkomma
<u>_TypeInt</u>	Ganzzahlig (32 Bit)
<u>_TypeLogic</u>	Logisch
<u>_TypeTime</u>	Zeit

Ist die Eigenschaft nicht vorhanden, wird 0 zurück gegeben.

### Beispiel

```
sub GetUserProp( aHdlUser : handle; aProperty : alpha; ) : alpha; local { tReturnValue
```

## Kontakt

obj -> UrmRead(int1, int2[, alpha3]) : alpha           Objekt des

Benutzer- und Rechtesystems lesen

obj Deskriptor des Eltern-Objekts oder 0

zu lesender Typ (abhängig von

handle1)

\_UrmTypeUser Benutzer lesen

\_UrmTypeUserGroup Benutzergruppe

lesen  
\_UrmTypeElmGroup Elementgruppe

lesen  
\_UrmTypeSysProperty Vordefinierte

int1 \_UrmTypeProperty Eigenschaft lesen  
Benutzerdefinierte

Eigenschaft lesen  
\_UrmTypeMember Mitglied oder  
Benutzergruppe

lesen  
\_UrmTypePerm Recht lesen  
\_UrmTypeElm... Element lesen

Lesemodus

0 Das angegebene  
Objekt (alpha4)

lesen  
\_UrmFirst erstes Objekt lesen

int2 \_UrmPrev vorheriges Objekt

lesen  
\_UrmNext nachfolgendes

Objekt lesen  
\_UrmLast letztes Objekt lesen

alpha3 Objektname (optional)

Resultat alpha Name des gelesenen Objekts 

Siehe Verwandte Befehle, Benutzerpflege

Dieser Befehl wird verwendet, um Objekte aus der Benutzerverwaltung zu lesen. In Abhängigkeit des übergebenen Objekttyps werden unterschiedliche Informationen ausgelesen. Folgende Informationen können gelesen werden:

- Benutzer

Um Benutzer zu lesen, wird als Eltern-Objekt (obj) 0 und in (int1)

\_UrmTypeUser übergeben. Der Rückgabewert ist der Anmeldename des Benutzers.

- Benutzergruppen

Um die Benutzergruppen zu lesen, wird als Eltern-Objekt (obj) 0 und in (int1)

\_UrmTypeUserGroup übergeben. Der Rückgabewert entspricht dem Namen der Benutzergruppe.

- Elementgruppe

## Kontakt

Um die Elementgruppen zu lesen, wird als Eltern-Objekt (obj) 0 und in (int1) UrmTypeElmGroup übergeben. Zurückgegeben wird der Name der Elementgruppe.

- **Eigenschaften**

Um die Eigenschaften zu lesen, wird als Eltern-Objekt (obj) das auszulesende Objekt übergeben. Soll eine von CONZEPT 16 vordefinierte Eigenschaft gelesen werden muss in (int1) UrmTypeSysProperty, soll eine benutzerdefinierte Eigenschaft gelesen werden, muss UrmTypeProperty übergeben werden. Zurückgegeben wird der Name der Eigenschaft.

- **Mitgliedschaften**

Um die Benutzer einer Benutzergruppe zu lesen, wird als Eltern-Objekt (obj) der Deskriptor der Benutzergruppe angegeben. Sollen die Benutzergruppen ermittelt werden, denen ein Benutzer angehört, wird als (obj) der Deskriptor des Benutzers angegeben. In (int1) wird UrmTypeMember übergeben.

Zurückgegeben wird der Name des Benutzers bzw. der Benutzergruppe.

- **Rechte einer Benutzergruppe**

Um die Elementgruppen mit bestimmten Berechtigungen zu lesen, wird der Deskriptor der Benutzergruppe als (obj) und in (int1) UrmTypePerm übergeben. Das Recht steht in (alpha3). Zurückgegeben werden alle Elementgruppen, auf die Rechte vergeben sind.

- **Eintrag in einer Elementgruppe**

Um die Einträge einer Elementgruppe zu lesen, wird als Eltern-Objekt (obj) der Deskriptor der Elementgruppe und in (int1) eine UrmTypeElm...-Konstante übergeben. Zurückgegeben wird der Name eines Eintrags.

In (int2) wird der Lesemodus übergeben. Soll das erste (UrmFirst) oder das letzte Element (UrmLast) gelesen werden, muss kein Referenz-Element in (alpha3) angegeben werden. Dies ist nur notwendig, wenn die Parameter UrmPrev oder UrmNext oder die Existenz eines Elements (int2 = 0) überprüft werden soll.

Zurückgegeben wird der Name des entsprechenden Elements. Konnte kein Objekt gefunden werden, wird ein Leerstring zurückgegeben.

Beispiele:

```
// Alle Benutzergruppen ermittelnfor tUsgGrpName # UrmRead(0, _UrmTypeUserGroup, _UrmFirst);loo
```

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig.
<u>ErrValueInvalid</u>	Der in (int1) angegebene Typ ist ungültig.
<u>ErrStringOverflow</u>	Der Objektname in (alpha3) ist zu lang.

Konstanten für Benutzerbefehle

Konstanten für Benutzerbefehle

Siehe Benutzerbefehle

- UrmAllow
- UrmDeny
- UrmFirst
- UrmIdePermCreate
- UrmIdePermDelete
- UrmIdePermModify
- UrmIdePermRead
- UrmLast
- UrmLock
- UrmNext
- UrmOldPermAccess
- UrmOldPermDelete
- UrmOldPermEntry
- UrmOldPermExecLists
- UrmOldPermExecSelections
- UrmOldPermExecTransfers
- UrmOldPermLink
- UrmOldPermListFormats
- UrmOldPermModify
- UrmOldPermParameters
- UrmOldPermRecLists
- UrmOldPermSave
- UrmOldPermSelections
- UrmOldPermTextMix
- UrmOldPermTransfers
- UrmPermConfig
- UrmPermCreate
- UrmPermDelete
- UrmPermDeleteOwner
- UrmPermElmGroupDelete
- UrmPermElmGroupInsert
- UrmPermElmGroupRead
- UrmPermExecute
- UrmPermMemberDelete
- UrmPermMemberInsert
- UrmPermModify
- UrmPermModifyOwner
- UrmPermRead
- UrmPermUser
- UrmPrev
- UrmStrict
- UrmTypeElmBlob
- UrmTypeElmCustom
- UrmTypeElmDialog
- UrmTypeElmElmGroup
- UrmTypeElmGroup
- UrmTypeElmMenu

- UrmTypeElmMetaPicture
- UrmTypeElmPicture
- UrmTypeElmPrintDocRecord
- UrmTypeElmPrintDocument
- UrmTypeElmPrintForm
- UrmTypeElmPrintFormList
- UrmTypeElmProcedure
- UrmTypeElmTable
- UrmTypeElmTheme
- UrmTypeElmUser
- UrmTypeElmUserGroup
- UrmTypeMember
- UrmTypePerm
- UrmTypeProperty
- UrmTypeSysProperty
- UrmTypeUser
- UrmTypeUserGroup

## Kontakt

**\_PwdModify**

**Passwort ändern**

**Wert 2 / 0x02**

**Verwandte**

**Siehe Befehle,**

**UserPassword()**

**Option bei UserPassword() durch die das Passwort verändert werden kann.**

## Kontakt

**\_PwdVerify**

**Passwort überprüfen**

**Wert 1 / 0x01**

**Verwandte**

**Siehe Befehle,**

**UserPassword()**

**Option bei UserPassword() durch die das Passwort überprüft werden kann.**



**\_UrmAllow**

**Recht zusichern**

**Wert 1**

**UrmPermGet()**,

Siehe **UrmPermGetRaw()**,

**UrmPermSet()**

Wird dieser Parameter bei dem Befehl **UrmPermSet()** angegeben, werden die übergebenen Rechte dem Benutzer oder der Benutzergruppe zugesichert.

Bei der Abfrage von Rechten mit den Anweisungen **UrmPermGet()** und **UrmPermGetRaw()** wird das Ergebnis nur für die zugesicherten Rechte zurückgegeben.

**\_UrmDeny**

**Recht entziehen**

**Wert -1**

**UrmPermGet(),**

**Siehe UrmPermGetRaw(),**

**UrmPermSet()**

**Wird dieser Parameter bei dem Befehl UrmPermSet() angegeben, werden die übergebenen Rechte dem Benutzer oder der Benutzergruppe entzogen.**

**Bei der Abfrage von Rechten mit den Anweisungen UrmPermGet() und UrmPermGetRaw() wird das Ergebnis nur für die entzogenen Rechte zurückgegeben.**

## Kontakt

**\_UrmFirst**

**Erstes Objekt lesen**

**Wert**<sup>1/</sup>

**0x00000001**

**Verwandte**

**Siehe Befehle,**

**UrmRead()**

**Option beim Befehl UrmRead() - das erste Objekt wird gelesen.**

**\_UrmIdePermCreate**

**Berechtigung zum Anlegen (Entwicklungsumgebung)**

Wert 2097152 /

**0x00200000**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet(),**

**UrmPermGetRaw(),**

**UrmPermSet()**

Mit dieser Konstante kann das Recht zum Anlegen in der Entwicklungsumgebung gesetzt oder entzogen werden.

Das betrifft das Anlegen von Datensätzen in der Datensatzverwaltung im Designer und der Standardverwaltung in der textbasierten Oberfläche, sowie das Anlegen von **binären Objekten** in der **BLOB-Verwaltung**.

Der Zugriff innerhalb der Laufzeitumgebung wird durch dieses Recht nicht eingeschränkt.



Dieses Recht ist gleichbedeutend mit **\_UrmOldPermSave.**

Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

**\_UrmIdePermDelete**

**Berechtigung zum Löschen (Entwicklungsumgebung)**

Wert 8388608 /

**0x00800000**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet()**,

**UrmPermGetRaw()**,

**UrmPermSet()**

Mit dieser Konstante kann das Recht zum Löschen in der Entwicklungsumgebung gesetzt oder entzogen werden.

Das betrifft das Löschen von Datensätzen in der Datensatzverwaltung im Designer und der Standardverwaltung in der textbasierten Oberfläche, sowie das Löschen von **binären Objekten** in der **BLOb-Verwaltung**.

Der Zugriff innerhalb der Laufzeitumgebung wird durch dieses Recht nicht eingeschränkt.



Dieses Recht ist gleichbedeutend mit **\_UrmOldPermDelete**.

Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

**\_UrmIdePermModify**

**Berechtigung zum Ändern (Entwicklungsumgebung)**

Wert 4194304 /

**0x00400000**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet()**,

**UrmPermGetRaw()**,

**UrmPermSet()**

Mit dieser Konstante kann das Recht zum Ändern in der Entwicklungsumgebung gesetzt oder entzogen werden.

Das betrifft das Ändern von Datensätzen in der Datensatzverwaltung im Designer und der Standardverwaltung in der textbasierten Oberfläche, sowie das Ändern von binären Objekten in der BLOB-Verwaltung.

Der Zugriff innerhalb der Laufzeitumgebung wird durch dieses Recht nicht eingeschränkt.



Dieses Recht ist gleichbedeutend mit **\_UrmOldPermModify**.

Dieses Recht kann bei den Standard-Elementgruppen \_blob und \_table gesetzt werden.

**\_UrmIdePermRead**

**Berechtigung zum Lesen (Entwicklungsumgebung)**

Wert 1048576 /

**0x00100000**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet(),**

**UrmPermGetRaw(),**

**UrmPermSet()**

Mit dieser Konstante kann das Recht zum Lesen in der Entwicklungsumgebung gesetzt oder entzogen werden.

Das betrifft das Lesen von Datensätzen in der Datensatzverwaltung im Designer und der Standardverwaltung in der textbasierten Oberfläche, sowie das Lesen von **binären Objekten** in der **BLOB-Verwaltung**.

Der Zugriff innerhalb der Laufzeitumgebung wird durch dieses Recht nicht eingeschränkt.



Dieses Recht ist gleichbedeutend mit **\_UrmOldPermAccess**.

Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

## Kontakt

**\_UrmLast**

**Letztes Objekt lesen**

Wert<sup>2/</sup>

**0x00000002**

**Verwandte**

Siehe **Befehle**,

**UrmRead()**

Option beim Befehl **UrmRead()** - das letzte Objekt wird gelesen.



## Kontakt

**\_UrmLock**

**Benutzerobjekt sperren**

Wert <sup>8/</sup>

**0x00000008**

**Verwandte**

Siehe **Befehle**,

**UrmOpen()**

Option beim Befehl **UrmRead()** - das Objekt wird gesperrt.

Bei dieser Sperre wird das Objekt benutzerbezogen gesperrt. D. h. bei einem weiteren Versuch vom gleichen Benutzer dieses Objekt zu sperren, wird kein Fehlerwert zurück gegeben.

Die Sperre bleibt bis zum Schließen des Objekts mit **UrmClose()** erhalten.

## Kontakt

**\_UrmNext**

**Nächstes Objekt lesen**

Wert <sup>4/</sup>

**0x00000004**

**Verwandte**

Siehe **Befehle**,

**UrmRead()**

Option beim Befehl **UrmRead()** - das nächste Objekt wird gelesen. Damit das nächste Objekt gelesen werden kann, muss ein Referenzobjekt angegeben werden.

**\_UrmOldPermAccess**

**Dateiberechtigung zum Zugriff auf Datensätze**

Wert 1048576 /

**0x00100000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Zugriff auf Datensätze" ab.

Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.



Dieses Recht ist gleichbedeutend mit **\_UrmIdePermRead.**

Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

**\_UrmOldPermDelete**

**Dateiberechtigung zum Löschen von Datensätzen**

Wert 8388608 /

**0x00800000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Löschen von Datensätzen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.



Dieses Recht ist gleichbedeutend mit **\_UrmIdePermDelete**.  
Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

**\_UrmOldPermEntry**

**Dateiberechtigung zur Eingabe von Datensätzen**

Wert 524288 /

**0x00080000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Eingabe von Datensätzen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.

Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.

**\_UrmOldPermExecLists**

**Dateiberechtigung Ausführen von Listen**

Wert 33554432 /

**0x02000000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

**Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ausführen von Listen" ab.**

**Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.**

**Dieses Recht kann bei der Standard-Elementgruppe \_table gesetzt werden.**

**\_UrmOldPermExecSelections Dateiberechtigung**

**Ausführen von Selektionen Wert 67108864 /**

**0x04000000**

**Verwandte**

**Befehle,**

**Konvertierung  
des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

**Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ausführen von Selektionen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.**

**Dieses Recht kann bei der Standard-Elementgruppe \_table gesetzt werden.**

**\_UrmOldPermExecTransfers**

**Dateiberechtigung zur Ausführung von Transfers**

Wert 134217728 /  
0x08000000

**Verwandte**

**Befehle,**

**Konvertierung  
des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ausführung von Transfers" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.

Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.



**\_UrmOldPermLink**

**Dateiberechtigung zum Zugriff auf Verknüpfungen**

Wert 262144 /

**0x00040000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

**Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Zugriff auf Verknüpfungen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.**

**Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.**

**\_UrmOldPermListFormats**

**Dateiberechtigung zur Änderung von Listenformaten**

Wert 536870912 /  
0x20000000

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ändern von Listenformaten" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.

Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.

**\_UrmOldPermModify**

**Dateiberechtigung zum Ändern von Datensätzen**

Wert 4194304 /

**0x00400000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ändern von Datensätzen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.



Dieses Recht ist gleichbedeutend mit **\_UrmIdePermModify**.  
Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

**\_UrmOldPermParameters**

**Dateiberechtigung zur Änderung von Dateiparametern**

Wert 16777216 /

**0x01000000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ändern von Dateiparametern" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.

Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.

**\_UrmOldPermRecLists**

Dateiberechtigung zum Ändern von Zugriffslisten Wert

**1073741824** /  
**0x40000000**

Verwandte

Befehle,

Konvertierung  
des alten

Siehe

Benutzersystems,

UrmPermGet(),

UrmPermGetRaw(),

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ändern von Zugriffslisten" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.

Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.

**\_UrmOldPermSave**

**Dateiberechtigung zum Speichern von Datensätzen**

Wert 2097152 /

**0x00200000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Speichern von Datensätzen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.



Dieses Recht ist gleichbedeutend mit **\_UrmIdePermCreate.**

Dieses Recht kann bei den **Standard-Elementgruppen** **\_blob** und **\_table** gesetzt werden.

**\_UrmOldPermSelections**

**Dateiberechtigung zum Ändern von Selektionen**

**Wert 268435456 /  
0x10000000**

**Verwandte**

**Befehle,**

**Konvertierung  
des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

**Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ändern von Selektionen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.**

**Dieses Recht kann bei der Standard-Elementgruppe \_table gesetzt werden.**

**\_UrmOldPermTextMix**

**Dateiberechtigung zum Text und Daten mischen**

**Wert 131072 /**

**0x00020000**

**Verwandte**

**Befehle,**

**Konvertierung**  
**des alten**

Siehe

**Benutzersystems,**

**UrmPermGet(),**

**UrmPermGetRaw(),**

**Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Daten und Text mischen" ab. Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.**

**Dieses Recht kann bei der Standard-Elementgruppe \_table gesetzt werden.**



**\_UrmOldPermTransfers**

Dateiberechtigung zum Ändern von Transfers Wert <sup>-</sup>  
**2.147.483.648 /**  
**0x80000000**

Verwandte

Befehle,

Konvertierung  
des alten

Siehe

Benutzersystems,

UrmPermGet(),

UrmPermGetRaw(),

Dieses Recht bildet aus dem alten Benutzersystem die Dateiberechtigung "Ändern von Transfers" ab.

Mit dieser Konstante kann das Recht gesetzt oder entzogen werden.

Dieses Recht kann bei der Standard-Elementgruppe `_table` gesetzt werden.

## Kontakt

**\_UrmPermConfig**

**Berechtigung zum Ändern der Datenstruktur Wert**

**128 / 0x00000080**

**Verwandte**

**Befehle,**

**Siehe UrmPermGet(),**

**UrmPermGetRaw(),**

**UrmPermSet()**

**Mit dieser Konstante kann das Recht zum Ändern der Datenstruktur gesetzt oder entzogen werden.**

**Dieses Recht kann bei de Standard-Elementgruppe \_table gesetzt werden.**

**\_UrmPermCreate**

**Berechtigung zum Anlegen**

**Wert 4 / 0x00000004**

**Verwandte**

Siehe Befehle,

**UrmPermGet()**,

**UrmPermSet()**

Mit dieser Konstante kann das Standard-Recht zum Anlegen gesetzt oder entzogen werden.

Dieses Recht kann bei den Standard-Elementgruppen \_dialog, \_elementgroup, \_menu, \_metapicture, \_picture, \_printdocrecord, \_printdocument, \_printform, \_printformlist, \_procedure, \_table, \_text, \_theme, \_user und \_usergroup gesetzt werden.

**\_UrmPermDelete**

**Berechtigung zum Löschen**

Wert <sup>16</sup> /

**0x00000010**

**Verwandte**

Siehe Befehle,

**UrmPermGet()**,

**UrmPermSet()**

Mit dieser Konstante kann das Standard-Recht zum Löschen gesetzt oder entzogen werden.

Dieses Recht kann bei den Standard-Elementgruppen **\_dialog**, **\_elementgroup**, **\_menu**, **\_metapicture**, **\_picture**, **\_printdocrecord**, **\_printdocument**, **\_printform**, **\_printformlist**, **\_procedure**, **\_table**, **\_text**, **\_theme**, **\_user** und **\_usergroup** gesetzt werden.

**\_UrmPermDeleteOwner**

**Berechtigung zum Löschen bei Objektbesitz**

Wert <sup>64</sup> /

**0x00000040**

**Verwandte**

Siehe Befehle,

**UrmPermGet()**,

**UrmPermSet()**

Mit dieser Konstante kann das Standard-Recht zum Löschen bei Objektbesitz gesetzt oder entzogen werden. Das Recht hat nur dann Auswirkung, wenn der Benutzer der Besitzer des entsprechenden Objekts der Elementgruppe ist.

Dieses Recht kann bei den Standard-Elementgruppen **\_user** und **\_usergroup** gesetzt werden.

## Kontakt

**\_UrmPermElmGroupDelete**

**Berechtigung zum Entfernen eines Elements aus einer Elementgruppe**

Wert 1024 /

**0x00000400**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet()**,

**UrmPermGetRaw()**,

**UrmPermSet()**

Mit dieser Konstante kann das Recht ein Element aus einer Elementgruppe zu entfernen zugesichert oder entzogen werden.

Dieses Recht kann nicht bei den Standard-Elementgruppen gesetzt werden.

## Kontakt

**\_UrmPermElmGroupInsert**

Berechtigung zum Einfügen von Elementen in eine Elementgruppe Wert 512 /  
0x00000200

Verwandte

Befehle,

Siehe UrmPermGet(),

UrmPermGetRaw(),

UrmPermSet()

Mit dieser Konstante kann das Recht ein Element einer Elementgruppe hinzuzufügen zugesichert oder entzogen werden.

Dieses Recht kann nicht bei den Standard-Elementgruppen gesetzt werden.

**\_UrmPermElmGroupRead**

**Berechtigung Elementgruppe lesen**

**Wert 256 / 0x00000100**

**Verwandte**

**Befehle,**

**Siehe UrmPermGet(),**

**UrmPermGetRaw(),**

**UrmPermSet()**

**Mit dieser Konstante kann das Recht eine Elementgruppe zu lesen zugesichert oder entzogen werden.**

**Dieses Recht kann bei der Standard-Elementgruppe \_elementgroup gesetzt werden.**



## Kontakt

**\_UrmPermExecute**

**Berechtigung zum Ausführen**

**Wert 1 / 0x00000001**

### **Verwandte**

Siehe Befehle,

**UrmPermGet()**,

**UrmPermSet()**

Mit dieser Konstante kann das Standard-Recht zum Ausführen gesetzt oder entzogen werden.

Dieses Recht kann bei den Standard-Elementgruppen \_dialog, \_menu und \_procedure gesetzt werden.

### **\_UrmPermMemberDelete**

**Berechtigung zum Entfernen von Benutzern aus einer Benutzergruppe**

Wert 8192 /

**0x00002000**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet()**,

**UrmPermGetRaw()**,

**UrmPermSet()**

Mit dieser Konstante kann das Recht einen Benutzer aus einer Benutzergruppe zu entfernen zugesichert oder entzogen werden.



Um einen Benutzer aus einer Benutzergruppe zu entfernen muss das Recht

\_UrmPermMemberDelete auf das entsprechende Benutzer- und Benutzergruppenobjekt erteilt werden.

Dieses Recht kann bei den Standard-Elementgruppen \_user und \_usergroup gesetzt werden.

### **\_UrmPermMemberInsert**

Berechtigung zum Einfügen von Benutzern in eine Benutzergruppe

Wert 4096 /

**0x00001000**

**Verwandte**

**Befehle,**

Siehe **UrmPermGet()**,

**UrmPermGetRaw()**,

**UrmPermSet()**

Mit dieser Konstante kann das Recht einen Benutzer einer **Benutzergruppe** hinzuzufügen zugesichert oder entzogen werden.



Um einen Benutzer einer Benutzergruppe hinzuzufügen muss das Recht

**\_UrmPermMemberInsert** auf das entsprechende **Benutzer-** und

**Benutzergruppenobjekt** erteilt werden.

Dieses Recht kann bei den **Standard-Elementgruppen** **\_user** und **\_usergroup** gesetzt werden.

**\_UrmPermModify**

**Berechtigung zum Ändern**

**Wert 8 / 0x00000008**

### Verwandte

Siehe Befehle,

UrmPermGet(),

UrmPermSet()

Mit dieser Konstante kann das Standard-Recht zum Ändern gesetzt oder entzogen werden.

Dieses Recht kann bei den Standard-Elementgruppen \_dialog, \_elementgroup, \_menu, \_metapicture, \_picture, \_printdocrecord, \_printdocument, \_printform, \_printformlist, \_procedure, \_table, \_text, \_theme, \_user und \_usergroup gesetzt werden.

**\_UrmPermModifyOwner**

**Berechtigung zum Ändern bei Objektbesitz**

Wert <sup>32 /</sup>

**0x00000020**

**Verwandte**

Siehe Befehle,

**UrmPermGet()**,

**UrmPermSet()**

Mit dieser Konstante kann das Standard-Recht zum Ändern bei Objektbesitz gesetzt oder entzogen werden. Das Recht hat nur dann Auswirkung, wenn der Benutzer der Besitzer des entsprechenden Objekts der Elementgruppe ist.

Dieses Recht kann bei den Standard-Elementgruppen **\_user** und **\_usergroup** gesetzt werden.

**\_UrmPermRead**

**Berechtigung zum Lesen**

**Wert 2 / 0x00000002**

**Verwandte**

Siehe Befehle,

UrmPermGet(),

UrmPermSet()

Mit dieser Konstante kann das Standard-Recht zum Lesen gesetzt oder entzogen werden.

Dieses Recht kann bei den Standard-Elementgruppen \_dialog, \_elementgroup, \_menu, \_metapicture, \_picture, \_printdocrecord, \_printdocument, \_printform, \_printformlist, \_procedure, \_table, \_text, \_theme, \_user und \_usergroup gesetzt werden.

**\_UrmPermUser**

**Benutzerdefinierte Berechtigung**

Verwandte

Befehle,

Siehe UrmPermGet(),

UrmPermGetRaw(),

UrmPermSet()

Zur Definition von benutzerdefinierten Rechten stehen 24 Konstanten zur Verfügung. Die Rechte werden als Bits in einem bigint abgelegt. Ein gesetztes Bit an einer bestimmten Stelle besagt, dass das Recht vergeben ist. Von CONZEPT 16 werden die niederwertigen 40 Bit verwendet. Die höherwertigen Bits können durch den Programmierer für eigene Rechte verwendet werden. Für die 24 höchstwertigen Bits sind folgende Konstanten definiert:

**\_UrmPermUser01**

**\_UrmPermUser02**

**\_UrmPermUser03**

...

**\_UrmPermUser23**

**\_UrmPermUser24**

Mit diesen Konstanten können einfach eigene Rechte gesetzt und abgefragt werden.

**Beispiel**

```
define{ _UrmPermPrint : _UrmPermUser01} ... tHdlUser->UrmPermSet('Customer', _UrmPermPrint | _
```

## Kontakt

**\_UrmPrev**

**Vorhergehendes Objekt lesen**

Wert<sup>3/</sup>

**0x00000003**

**Verwandte**

Siehe **Befehle**,

**UrmRead()**

Option beim Befehl **UrmRead()** - das vorhergehende Objekt wird gelesen. Damit das Objekt gelesen werden kann, muss ein Referenzobjekt angegeben werden.



## Kontakt

**\_UrmStrict**

**Sonderrecht für eigene Benutzerkomponenten nicht berücksichtigen Wert 2048**

**Siehe UrmOpen()**

**Ein Benutzer hat immer das Recht, eigene Eigenschaften zu lesen und zu verändern, sowie Eigenschaften seiner Benutzergruppen zu lesen. Durch die Angabe von \_UrmStrict bei UrmOpen() wird dieses Sonderrecht nicht berücksichtigt.**

## Kontakt

**\_UrmTypeElmGroup**

**Elementgruppe**

**Wert 3**

**Verwandte**

**Befehle,**

**Siehe UrmRead(),**

**UrmCreate(),**

**UrmDelete()**

**Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf eine Elementgruppe.**

## Kontakt

**\_UrmTypeMember**

**Mitgliedschaft**

**Wert 6**

**Verwandte**

**Befehle,**

**Siehe UrmRead(),**

**UrmCreate(),**

**UrmDelete()**

**Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf eine Mitgliedschaft in einer Benutzergruppe bzw. die Benutzergruppen in denen ein Benutzer Mitglied ist.**

## Kontakt

**\_UrmTypePerm**

**Berechtigungen in der Benutzerpflege**

**Wert 7**

**Verwandte**

**Befehle,**

**Siehe UrmRead(),**

**UrmCreate(),**

**UrmDelete()**

**Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf die Rechte einer Benutzergruppe.**

## Kontakt

\_UrmTypeProperty

Benutzerdefinierte Eigenschaft in der Benutzerpflege Wert 5

Verwandte Befehle,

\_UrmTypeSysProperty,

Siehe UrmRead(),

UrmCreate(),

UrmDelete()

Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf eine benutzerdefinierte Eigenschaft. Von CONZEPT 16 vorgegebene Eigenschaften werden mit \_UrmTypeSysProperty angesprochen.

\_UrmTypeSysProperty

Systemeigenschaft in der Benutzerpflege

Wert 4

Verwandte

Befehle,

Siehe \_UrmTypeProperty,

UrmRead(),

UrmCreate(),

UrmDelete()

Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf eine Eigenschaft, die von CONZEPT 16 vorgegeben ist. Benutzerdefinierte Eigenschaften werden mit \_UrmTypeProperty angesprochen.

## Kontakt

**\_UrmTypeUser**

**Benutzer**

**Wert 1**

**Verwandte**

**Befehle,**

**Siehe UrmRead(),**

**UrmCreate(),**

**UrmDelete()**

**Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf einen Benutzer.**

## Kontakt

**\_UrmTypeUserGroup**

**Benutzergruppe**

**Wert 2**

**Verwandte**

**Befehle,**

**Siehe UrmRead(),**

**UrmCreate(),**

**UrmDelete()**

**Option bei den Befehlen UrmRead(), UrmCreate() und UrmDelete(). Die entsprechenden Befehle beziehen sich auf eine Benutzergruppe.**



## Kontakt



**UserClear(int1, int2, alpha3) : int**  
**Datenbankbenutzer abmelden int1**  
**Benutzer-ID**  
**int2 Benutzernummer alpha3**  
**Server-Kennwort**

**Abmelderesultat**

**ErrOk**

**ErrDbidUserInvalid**

**Resultat int**

**ErrDbidUserSelf**

**ErrDbidAreaPassword Server-Kennwort**

**Abmelden**  
**erfolgreich**  
**Benutzer nicht**  
**eingeloggt oder**  
**Benutzer-ID (int1)**  
**passt nicht zu**  
**Benutzernummer**  
**(int2)**  
**Eigene**  
**Benutzer-ID in**  
**(int1) angegeben,**  
**Abmelden nicht**  
**möglich**  
**Server-Kennwort**  
**(alpha3) falsch**

**Siehe Verwandte Befehle, UserInfo()**

**Mit diesem Befehl wird ein Datenbankbenutzer zwangsweise aus der Datenbank ausgeloggt. Die benötigten Informationen können mit dem Befehl UserInfo() ermittelt werden.**

**In (int1) wird die Benutzer-ID, in (int2) die Benutzernummer des zu entfernenden Benutzers und in (alpha3) wird das Serverkennwort übergeben. Ist der Server nicht durch ein weiteres Kennwort geschützt, wird ein Leerstring (") übergeben.**



**Der Benutzer ist nicht sofort abgemeldet. Es kann bis zu zehn Sekunden dauern, bis der Benutzer tatsächlich beim Server ausgetragen ist.**

## Kontakt

UserCreate(alpha1, alpha2[,  
alpha3]) : int



Datenbankbenutzer anlegen

alpha1 Name des Benutzers

alpha2 Hauptbenutzergruppe

alpha3 Passwort (optional)

Resultat des Anlegevorgangs

ErrOk

Anlegen des  
Benutzers

erfolgreich

ErrUrmObjectNotFound Die

Benutzergruppe  
in (alpha2) ist  
nicht vorhanden

Resultat int ErrExists

Der neue  
Benutzer  
(alpha1) ist  
bereits

vorhanden

ErrRights

Der aktuelle  
Benutzer hat  
keine  
ausreichende  
Berechtigung

Siehe Verwandte Befehle, UrmCreate(), UrmDelete()



Dieser Befehl wurde durch UrmCreate() abgelöst und sollte nicht mehr verwendet werden.

Mit diesem Befehl kann ein neuer Datenbankbenutzer in der geöffneten Datenbank angelegt werden. Der Name des neuen Benutzers wird in (alpha1) übergeben. In (alpha2) wird die Hauptbenutzergruppe übertragen. In (alpha3) kann schließlich optional ein Passwort für den neuen Benutzer definiert werden.

Mögliche Laufzeitfehler:

ErrStringOverflow Alphanumerischer Wert zu lang

ErrValueInvalid

Bei einer Eingabeüberprüfung wurde ein ungültiger Wert erkannt

## Kontakt



UserDelete(alpha1) : int

Datenbankbenutzer löschen

alpha1 Name des Benutzers

Resultat des Löschvorgangs

ErrOk

Löschen des  
Benutzers

erfolgreich  
Der Benutzer  
(alpha1) ist  
gesperrt

ErrLocked

Resultat int ErrUrmObjectNotFound

Der Benutzer  
(alpha1) ist  
nicht

ErrRights

vorhanden  
Der aktuelle  
Benutzer hat  
keine  
ausreichende  
Berechtigung

Siehe

Verwandte Befehle, UrmCreate(),

UrmDelete()



Dieser Befehl wurde durch UrmDelete() abgelöst und sollte nicht mehr verwendet werden.

Mit diesem Befehl kann der Datenbankbenutzer (alpha1) aus der geöffneten Datenbank gelöscht werden.

Mögliche Laufzeitfehler:

ErrStringOverflow Alphanumerischer Wert zu lang

ErrValueInvalid

Bei einer Eingabeüberprüfung wurde ein ungültiger Wert erkannt

**UserID(int1) : int**

**Benutzer-ID ermitteln**

**Informationstyp**

**UserCurrent Aktuellen**

**Benutzer**

**int1**

**ermitteln**

**UserLocked Sperrenden**

**Benutzer**

**ermitteln**

**Resultat int Benutzer-ID**

**Verwandte Befehle,**

**UserInfo(),**

Siehe

**UserNumber(),**

**UserName()**



Dieser Befehl wurde durch **UserInfo()** abgelöst und sollte nicht mehr verwendet werden.

Mit dieser Funktion kann die ID eines Benutzers ermittelt werden.

Folgende Optionen (int1) sind zulässig:

- **UserCurrent**

Es wird die ID des aktuellen Benutzers zurückgeliefert. Die ID ist ein Wert im Bereich zwischen 1 und 65535. Der Wert kann beispielsweise für die Generierung temporärer, benutzerbezogener Datensätze verwendet werden. Der Wert ist bei jedem Aufruf der Datenbank verschieden.

- **UserLocked**

Wird bei einem Zugriff in die Datenbank auf einen gesperrten Datensatz zugegriffen (Resultat = **rLocked**), so kann hiermit die ID des Benutzers ermittelt werden, der den Datensatz gesperrt hat. Gleiches gilt auch beim Zugriff auf Texte, Selektionen oder Listenformate. Der Wert bleibt solange erhalten, bis erneut auf einen gesperrten Satz, Text oder Parameter zugegriffen wird.

## Kontakt

**UserInfo(int1[, int2[, int3]]) :**



**alpha**

**Benutzerinformationen ermitteln**

int1 Informationstyp (siehe

Text)

int2 Benutzer-ID (optional)

int3 Verbundene Datenbank

(optional)

**Resultat alpha Benutzerinformation**

Verwandte Befehle,

UserClear(),

Siehe

UserPassword(),

DbInfo(), Beispiel

Mit diesem Befehl können Informationen über einen Benutzer ermittelt werden. Die Informationen liegen auf dem Server in einer Benutzertabelle vor, aus der einzelne Einträge geladen werden können. Das Übertragen der Benutzerinformationen erfolgt bei der Verwendung des Parameters UserCurrent, UserLocked, UserNextID, UserGroup oder wenn (int2) größer 0 ist. Erst im Anschluss daran können weitere Informationen des Benutzers abgefragt werden.

Um zum Beispiel den Namen des aktuellen Benutzers zu ermitteln, muss zunächst der Eintrag mit dem Befehl UserInfo(\_UserCurrent) abgeholt werden, bevor der Benutzername mit dem Befehl UserInfo(\_UserName) ermittelt werden kann. Sollen Informationen zu einem Benutzer mit einer bestimmten Benutzer-ID ermittelt werden, kann die Benutzer-ID im Parameter (int2) und die gewünschte Information in (int1) übergeben werden. Alle weiteren Informationen beziehen sich dann auf die zuletzt geladenen Benutzerinformationen. Ist kein Benutzer mit der angegebenen User-ID angemeldet, wird ein Leerstring zurückgegeben.

In (int3) kann angegeben werden, aus welcher verbundenen Datenbank (siehe DbConnect()) die angegebene Information ermittelt werden soll. Es können die Konstanten Db2 bis Db8 übergeben werden.



Die verbundene Datenbank kann nur bei den Optionen UserCurrent, UserLocked, UserNextID und UserGroup angegeben werden, da diese Informationen aus der Datenbank lesen.

Resultate, die numerische Informationen enthalten, müssen gegebenenfalls mit CnvIA() umgewandelt werden.



Die Befehle UserID(), UserNumber() und UserName() werden durch diesen Befehl ersetzt und sollten nicht mehr verwendet werden.

Von einem Benutzer können sowohl die Benutzer-ID als auch die Benutzer-Nummer ermittelt werden. Beide Nummern werden beim Anmelden des Benutzers durch CONZEPT 16 vergeben. Die Benutzer-Nummer startet immer bei 1. Der Benutzer, der sich zuerst anmeldet bekommt die Nummer 1, dann die 2 usw. Entstehende Lücken durch das Abmelden von Benutzern werden bei der Anmeldung neuer Benutzer wieder aufgefüllt.

Die Benutzer-ID wird in der Datenbank gespeichert und liegt im Bereich 1 bis 65535. Bei der Anmeldung eines Benutzers wird die gespeicherte Benutzer-ID erhöht und

## Kontakt

dem Benutzer zugewiesen. Ist der Maximalwert erreicht, beginnt die Nummerierung wieder bei 1. Die Benutzer-ID kann verwendet werden, wenn benutzer-eindeutige Namen benötigt werden, zum Beispiel für temporäre externe Dateien oder Namen für Selektionen.

Sowohl die Benutzer-Nummer als auch die Benutzer-ID sind eindeutig. Die Benutzer-Nummer wird aber nach dem Abmelden des Benutzers sofort wieder verwendet, die Wiederverwendung der Benutzer-ID findet erst nach 65535 neuen Anmeldungen statt. Innerhalb der Protokolldatei der Datenbank werden beim Login/Logout eines Benutzers dessen Benutzer-ID angegeben.

Folgende Optionen (int1) sind zulässig:

- UserCurrent

Der Eintrag des eigenen Benutzers wird in der Benutzertabelle gelesen und dessen Benutzer-ID zurückgegeben.

- UserLocked

Der Eintrag des sperrenden Benutzers wird gelesen und dessen Benutzer-ID zurückgegeben.

Diese ID kann nach einer Datenbankoperation mit den Ergebnissen rLocked und ErrBinLocked abgefragt werden. Diese Ergebnisse werden von Datensatzoperationen oder beim Zugriff auf Selektionen, Texte, binäre Objekte und Listenformate zurückgegeben.

- UserNextID

Mit dieser Option kann die Benutzer-ID des nächsten Benutzers ermittelt werden. In (int2) steht die Benutzer-ID des Tabelleneintrags, der zuvor gelesen wurde. Wird in (int2) 0 übergeben, wird der erste Benutzer in der Tabelle gelesen.

- UserName

Name des Benutzers.

- UserNumber

Nummer des Benutzers. Zur Unterscheidung zur Benutzer-ID siehe oben.

- UserProtocol

Der Rückgabewert entspricht dem Protokoll mit dem der Client und der Server verbunden sind. Es wird nur TCP unterstützt.

- UserAddress

Es wird die IP-Adresse (x.x.x.x) des Rechners zurückgegeben. Siehe auch NetInfo().

- UserSysName Name

des Rechners.

- UserSysNameIP

## Kontakt

IP-Name des Rechners.

Der Name setzt sich aus dem Rechnernamen und der Arbeitsgruppe zusammen.

- UserLoginDate

Es wird das Datum der letzten Anmeldung zurückgegeben.

- UserLoginTime

Es wird die Uhrzeit der letzten Anmeldung zurückgegeben.

- UserLogin

Es werden die Anzahl der Sekunden seit des Logins zurückgegeben.

- UserLastReqDate

Es wird das Datum zurückgegeben, zu dem der Benutzer zuletzt eine Anfrage an den Server geschickt hat.

- UserLastReqTime

Es wird die Uhrzeit zurückgegeben, zu dem der Benutzer zuletzt eine Anfrage an den Server geschickt hat.

- UserLastReq

Es wird die Anzahl der Sekunden seit der letzten Serveranfrage zurückgegeben.

- UserSysAccount

Name des Systembenutzers. Dies ist der Name des Windows- oder Unix-Benutzerkontos.

- UserNetAccount

Name des Netzwerkbenutzers. Dies ist der Name des Netware-Benutzerkontos.

- UserGroup

Hauptbenutzergruppe ermitteln. In (int2) muss die Benutzer-ID des Benutzers angegeben werden, dessen Hauptbenutzergruppe ermittelt werden soll. Es wird ein Leerstring zurückgegeben, wenn es sich bei dem Benutzer um keinen untergeordneten Benutzer handelt. In diesem Fall kann mit UrmPropGet( UrmPropUserGroup) die Hauptbenutzergruppe ermittelt werden.

- UserPlatform

Prozedurumgebung ermitteln. Rückgabewert ist eine \_Pfm...-Konstante.

- UserJobID

Job-ID ermitteln (siehe JobID).

Beispiele:

```
// Lesen aller Benutzer// Eintrag des ersten Benutzers beim Server abholenfor tID # CnvIA(UserI
```

UserName(int1) : alpha 

Benutzername ermitteln

ID eines beliebigen

Benutzers

UserCurrent ID des  
aktuellen  
Benutzers

int1

UserLocked ID des  
sperrenden  
Benutzers

Resultat alpha Benutzername

Verwandte Befehle,

Siehe UserInfo(), UserNumber(),  
UserID()



Dieser Befehl wurde durch UserInfo() abgelöst und sollte nicht mehr verwendet werden.

Mit dieser Funktion kann der Name eines Benutzers ermittelt werden.

In (int1) wird eine Benutzer-ID übergeben. Bei einer ungültigen Benutzer-ID ist das Resultat leer (''). Folgende Optionen (int1) sind ebenfalls zulässig:

- UserCurrent

Das Resultat ist der Name des aktuellen Benutzers

- UserLocked

Das Resultat ist der Name des sperrenden Benutzers



UserNumber(int1) : int        

Benutzernummer ermitteln

ID eines beliebigen

Benutzers

UserCurrent ID des  
aktuellen  
Benutzers

int1

UserLocked ID des  
sperrenden  
Benutzers

Resultat int Benutzernummer

Verwandte Befehle,

Siehe UserInfo(), UserName(),  
UserID()



Dieser Befehl wurde durch UserInfo() abgelöst und sollte nicht mehr verwendet werden.

Mit dieser Funktion kann die Nummer eines Benutzers ermittelt werden.

In (int1) wird eine Benutzer-ID übergeben. Bei einer ungültigen Benutzer-ID ist das Resultat 0. Folgende Optionen (int1) sind ebenfalls zulässig:

- UserCurrent

Das Resultat ist die Nummer des aktuellen Benutzers

- UserLocked

Das Resultat ist die Nummer des sperrenden Benutzers

## Kontakt

UserPassword(alpha1, alpha2,  
alpha3[, int4) : int



Benutzerpasswort ändern

alpha1 Benutzername

alpha2 altes Benutzerpasswort

alpha3 neues Benutzerpasswort

int4 Optionen (optional)

PwdModify Passwort ändern

PwdVerify Altes Passwort überprüfen

Änderungsergebnis

ErrOK

Änderung erfolgreich

ErrUrmParentNotFound Benutzer (alpha1) nicht

ErrRights vorhanden

Altes Benutzerpasswort

(alpha2) falsch oder

Benutzerrechte nicht

ausreichend

Resultat int ErrData

Neues Benutzerpasswort

(alpha3) entspricht nicht

den Passwortrichtlinien

(UrmPropPwdCapitals,

UrmPropPwdDigits,

UrmPropPwdLocked,

UrmPropPwdMinLength

oder

UrmPropPwdSpecials)

des Benutzers

Siehe Verwandte Befehle, UserInfo()

Dieser Befehl ändert das eigene Kennwort (alpha1 = leer oder der eigene

Benutzername) oder das Kennwort eines anderen Benutzers.

Es können folgende Optionen (int4) übergeben werden:

PwdModify Das Passwort wird verändert

PwdVerify Das Passwort wird überprüft



Aus Sicherheitsgründen verzögert sich die Passwortüberprüfung, wenn das Passwort eines Benutzers mehr als drei Mal mit einem falschen Passwort überprüft wird.

Die Optionen können kombiniert werden, so dass das alte Passwort überprüft wird, bevor es gesetzt wird.

Ist der Parameter Optionen (int4) nicht angegeben, muss beim Ändern des eigenen Kennworts das alte Kennwort angegeben werden. Wird das Kennwort eines anderen Benutzers geändert, muss das alte Kennwort nicht angegeben werden, stattdessen müssen aber die entsprechenden Änderungsrechte für den angegebenen Benutzer vorhanden sein.

## Kontakt



Wird in den Optionen 0 oder NULL, dann macht der Befehl nichts.

Beispiele:

```
// Passwort des eigenen Benutzers mit vorheriger Prüfung ändernUserPassword('', tOldPass, tNewPas  
// Passwort eines Benutzers verifizieren (z. B. Authentifizierung)if (UserPassword(tUser, tPass,
```



Das Passwort eines Benutzers kann auch mittels UrmPropSet() mit der Option UrmPropPassword gesetzt werden.

Mögliche Laufzeitfehler:

ErrStringOverflow Benutzername (alpha1), altes Benutzerkennwort (alpha2) oder  
neues Benutzerkennwort (alpha3) zu lang (max. 20 Zeichen  
zulässig)

Konstanten für Benutzerinformationsbefehle  
Konstanten für Benutzerinformationsbefehle Siehe  
Benutzerbefehle

- UserAddress
- UserCurrent
- UserGroup
- UserJobID
- UserLastReq
- UserLastReqDate
- UserLastReqTime
- UserLocked
- UserLogin
- UserLoginDate
- UserLoginTime
- UserName
- UserNetAccount
- UserNextID
- UserNumber
- UserPlatform
- UserProtocol
- UserSysAccount
- UserSysName
- UserSysNameIP

## Kontakt

**\_UserAddress**

Netzwerkadresse ermitteln

Wert 3

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die die Netzwerkadresse (IP-Adresse) des Clients ermittelt werden kann.

## Kontakt

**\_UserCurrent**

Aktuellen Benutzer ermitteln

Wert 0

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die die ID des aktuellen Benutzers ermittelt werden kann.

Die Benutzer-ID liegt im Bereich von 1 bis 65535.

**\_UserGroup**

**Hauptbenutzergruppe ermitteln**

**Wert 16**

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die der Name der Hauptbenutzergruppe ermittelt werden kann.



Handelt es sich bei dem Benutzer um keinen untergeordneten Benutzer, wird ein leeres Resultat zurückgegeben. In diesem Fall kann mit **UrmPropGet( UrmPropUserGroup)** die Hauptbenutzergruppe ermittelt werden.

## Kontakt

**\_UserJobID**

**JobID** ermitteln

Wert 18

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die die **JobID** eines Benutzers ermittelt werden kann.



## Kontakt

**\_UserLastReq**

Zeit seit letzter Anfrage ermitteln

Wert 11

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die die Anzahl der Sekunden seit der letzten Serveranfrage ermittelt werden kann.

## Kontakt

**\_UserLastReqDate**

Datum der letzten Anfrage ermitteln

Wert 12

Verwandte

Siehe Befehle,

UserInfo()

Option bei UserInfo() durch die das Datum der letzten Serveranfrage ermittelt werden kann.

## Kontakt

**\_UserLastReqTime**

Uhrzeit der letzten Anfrage ermitteln

Wert 13

Verwandte

Siehe Befehle,

UserInfo()

Option bei UserInfo() durch die die Uhrzeit der letzten Serveranfrage ermittelt werden kann.

## Kontakt

**\_UserLocked**

Sperrenden Benutzer ermitteln

Wert -1

Verwandte

Siehe Befehle,

UserInfo()

Option bei UserInfo() durch die die ID des sperrenden Benutzers ermittelt werden kann.

Wird bei einem Zugriff in die Datenbank auf einen gesperrten Datensatz zugegriffen (Resultat = rLocked), so kann hiermit die ID des Benutzers ermittelt werden, der den Datensatz gesperrt hat.

Ebenso kann beim Zugriff auf Texte, Selektionen oder binäre Objekte (Resultat = ErrBinLocked) der sperrende Benutzer ermittelt werden. Der Wert bleibt solange erhalten, bis erneut auf einen gesperrten Satz, Text oder Parameter zugegriffen wird.

## Kontakt

**\_UserLogin**

Zeit seit letzter Anmeldung ermitteln

Wert 8

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die die Anzahl der Sekunden seit der letzten Anmeldung ermittelt werden kann.

## Kontakt

**\_UserLoginDate**

Datum der letzten Anmeldung ermitteln

Wert 9

Verwandte

Siehe Befehle,

UserInfo()

Option bei UserInfo() durch die das Datum der letzten Anmeldung ermittelt werden kann.

## Kontakt

**\_UserLoginTime**

Uhrzeit der letzten Anmeldung ermitteln

Wert 10

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die die Uhrzeit der letzten Anmeldung ermittelt werden kann.

## Kontakt

**\_UserName**

**Benutzername ermitteln**

**Wert 4**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die der Name des Benutzers ermittelt werden kann. Wird als Name ? zurückgegeben, handelt es sich um einen Benutzer, der sich noch nicht authentifiziert hat, also noch bei der Anmeldung an der Datenbank ist.**



## Kontakt

**\_UserNetAccount**

**Netware-Benutzername ermitteln**

**Wert 15**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die der Name des Netware-Benutzers ermittelt werden kann.**

**Ist der Benutzer nicht bei einem Netware-Server angemeldet, ist das Resultat leer ('').**

## Kontakt

**\_UserNextID**

**Nächste Benutzer-ID ermitteln**

**Wert 1**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die die Nächste Benutzer-ID ermittelt werden kann.**

## Kontakt

**\_UserNumber**

**Benutzernummer ermitteln**

**Wert 7**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die die Nummer des Benutzers ermittelt werden kann.**

## Kontakt

**\_UserPlatform**

**Prozedurumgebung ermitteln**

**Wert 17**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die die Prozedurumgebung eines Benutzers ermittelt werden kann.**

**Es werden die \_Pfm...-Konstanten zurückgegeben.**

## Kontakt

**\_UserProtocol**

**Benutzerprotokoll ermitteln**

**Wert 2**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die das zur Kommunikation mit dem Server verwendete Protokoll des Benutzers ermittelt werden kann.**

**Es kann nur "TCP" zurückgegeben werden, da nur TCP/IP als Protokoll unterstützt wird.**

## Kontakt

**\_UserSysAccount**

System-Benutzername ermitteln

Wert 14

**Verwandte**

Siehe **Befehle**,

**UserInfo()**

Option bei **UserInfo()** durch die der Name des Windows- oder Unix-Benutzers ermittelt werden kann.

## Kontakt

**\_UserSysName**

**Rechnername ermitteln**

**Wert 5**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die der Name des Rechners ermittelt werden kann.**

**Unter Windows ist dies der NETBIOS-Name, bei Netware der Servername und bei Unix der Rechnername (uname).**

## Kontakt

**\_UserSysNameIP**

**Rechner-IP-Name ermitteln**

**Wert 6**

**Verwandte**

**Siehe Befehle,**

**UserInfo()**

**Option bei UserInfo() durch die der IP-Name des Rechners ermittelt werden kann.**

**Er wird entweder über die HOSTS-Tabelle oder über DNS ermittelt. Je nach Resolver ist die Domain-Information im Namen enthalten.**



### OEM-Kit-Befehle

Befehle des OEM-Kits  
Siehe Befehlsgruppen,

#### Befehlsliste



Die Anweisungen können nur ausgeführt werden, wenn die Datenbank mit dem CONZEPT 16-Standardclient (c16\_winc.exe) geöffnet wurde.

#### Befehle

- OEMLoad
- OEMSave

#### Konstanten

- OEMWait

## Kontakt

OEMLoad(alpha1[, alpha2[, var  
alpha3]]) : int



Datenbankdefinition laden

alpha1 Name der Definitionsdatei

alpha2 Post-Prozedur (optional)

var Detaillierte OEM-Kit-Fehlermeldung

alpha3 (optional)

Laderesultat

ErrOk

Laden

erfolgreich

Datenbank ist

mit dem

Advanced-Client

geöffnet

Datenbank

ErrOemDbalock

gesperrt

Definitionsdatei

ErrOemOpenFailed

kann nicht

geöffnet werden

ErrOemInvalidFormat Definitionsdatei

hat falsches

Format oder



wurde mit

anderer

CONZEPT

16-Version

erstellt

Nicht genügend

Speicher zum

Laden verfügbar

ErrOemOutOfSpace

ErrOemOpenDesigner Prozedur wurde

aus Designer

gestartet

Noch

mindestens ein

ErrOemOpenFrame

Dialog geöffnet

Resultat int

Siehe Verwandte Befehle, OEMSave()

Mit dieser Funktion wird eine Datenbankdefinition geladen. Beim Start der Funktion darf sich kein anderer Benutzer in der Datenbank befinden und es darf kein Dialog mehr offen sein. Die Funktion kann nicht innerhalb des Designers ausgeführt werden.



Die Anweisung kann nur ausgeführt werden, wenn die Datenbank mit dem CONZEPT 16-Standardclient (c16\_winc.exe) geöffnet wurde. Wurde der Advanced-Client (c16\_apgi.exe) verwendet, wird der Fehler ErrGeneric zurückgegeben.

Sofern mehrere Definitionsdateien vorhanden sind (.d02 usw.), müssen sich diese im selben Verzeichnis befinden, in dem sich auch die .d01-Datei befindet. Ein Datenträgerwechsel wird nicht unterstützt.

## Kontakt

Nach dem erfolgreichen Öffnen der Definitionsdatei und dem Entladen der Datenstruktur kann die Funktion nicht mehr zurückkehren. Tritt nach diesem Zeitpunkt ein Fehler auf, erfolgt eine entsprechende Bildschirmmeldung und der Client wird beendet. Sofern das Einlesen ohne Fehler durchgeführt wurde, wird anschliessend die in (alpha2) angegebene Prozedur gestartet. Ist diese Prozedur Bestandteil der Definitionsdatei, wird die neue Prozedur gestartet. In der Prozedur kann auf neue Strukturelemente, welche über die Definition eingelesen wurden, zugegriffen werden. Nach dem Ende der Prozedur wird der CONZEPT 16-Client beendet.

Existiert die in (alpha2) angegebene Prozedur nicht, wird nach dem Einlesen der Definition die Meldung "<Prozedurname>: Prozedur nicht vorhanden" ausgegeben. Nach Bestätigung der Meldung wird der CONZEPT 16-Client beendet.



Vor dem Einlesen eines Updates muss in jedem Falle eine Sicherung der Datenbank vorgenommen werden. Wird der Einlesevorgang unterbrochen (zum Beispiel durch Ausschalten des Rechners), kann die Datenbank beschädigt werden. Um möglichen Problemen beim Einlesen eines Updates vorzubeugen, sollte vor dem Update eine Diagnose der Datenbank durchgeführt werden.

Beispiel:

```
if (OEMLoad('C:\Cl6\Definition', '', var t.aErr) != _ErrOk){ WinDialogBox(0, 'Fehler', t.aErr, _
```

## Kontakt

OEMSave(alpha1, alpha2[, int3, var  
alpha4]) : int



Datenbankdefinition sichern

alpha1 Name der OEM-Kit-Gruppe

alpha2 OEM-Kit-Kennwort

Optionen (optional)

int3 \_OEMWait Statusanzeige nicht

automatisch schließen

var Detaillierte OEM-Kit-Fehlermeldung

alpha4 (optional)

Sicherungsresultat

\_ErrOk

Sichern erfolgreich

\_ErrGeneric

Datenbank ist mit dem  
Advanced-Client  
geöffnet

Resultat int \_ErrOemDbalock

Datenbank gesperrt

\_ErrOemOpenFailed Definitionsdatei kann

nicht geöffnet werden  
OEM-Kit-Passwort

\_ErrOemPassword

falsch

\_rLastRec

OEM-Kit-Gruppenname

nicht gefunden

Siehe Verwandte Befehle, OEMLoad(),

LangDisplay

Dieser Befehl exportiert die Datenbankdefinition der angegebenen Definitionsgruppe. Dabei findet kein erneuter Aufbau der Datenstruktur statt. Der Export ist nur bei vorhandenem OEM-Kit und der Angabe eines gültigen Kennworts möglich.



Die Anweisung kann nur ausgeführt werden, wenn die Datenbank mit dem CONZEPT 16-Standardclient (c16\_winc.exe) geöffnet wurde. Wurde der Advanced-Client (c16\_apgi.exe) verwendet, wird der Fehler \_ErrGeneric zurückgegeben.



Zum Zeitpunkt des Auslagerns darf nur ein Benutzer an der Datenbank angemeldet sein.

Beispiel:

```
if (OEMSave('UPDATE', 'PasswdOEM', 0, var t.aErr) != _ErrOk){ WinDialogBox(0, 'Fehler', t.aErr,
```

Eine so erstellte Definitionsdatei kann von einem CONZEPT 16-Client mit der gleichen Version in eine andere Datenbank eingelesen werden. Dazu steht entweder das entsprechende Menü in der Entwicklungsumgebung (Datenbank / Datensicherung / Datenbankdefinition einlesen) oder der Befehl OEMLoad() zur Verfügung.

Die Sprache, die in dem angezeigte Dialog verwendet wird, kann über die Eigenschaft LangDisplay gesteuert werden.

**Konstanten für OEM-Kit-Befehle**

**Konstanten für OEM-Kit-Befehle**

Siehe OEM-Kit-Befehle

- OEMWait

## Kontakt

**\_OEMWait**

Statusanzeige nicht automatisch schließen

Wert <sup>1/</sup>

**0x00000001**

**Verwandte**

Siehe **Befehle**,

**OEMSave()**

Option bei **OEMSave()** durch die ein automatisches Schließen der Statusanzeige nach dem Exportieren verhindert werden kann.

## Kontakt

### Validierungsbefehle

Liste der Befehle und Konstanten zur Validierung von Datenbankelementen

Befehlsgruppen,

Siehe Befehlsliste,

Validierungs-Editor

### Befehle

- VldClose
- VldDelete
- VldDirOpen
- VldDirRead
- VldOpen
- VldUpdate

### Konstanten

- VldCreate
- VldFirst
- VldLast
- VldLock
- VldNext
- VldPrev
- VldStateDeleted
- VldStateIrrelevant
- VldStateModified
- VldStateNotVerified
- VldStateUndefined
- VldStateVerified
- VldTypeDialog
- VldTypeMenu
- VldTypeProcedure
- VldTypeTable

## Kontakt

obj -> VldClose()  Validierungsverzeichnis oder Validierungselement  
schließen

Validierungsverzeichnis

obj oder

Validierungselement

Verwandte Befehle,

Siehe VldOpen(),

VldDirOpen()

Mit dieser Funktion wird das Validierungsverzeichnis oder das Validierungselement geschlossen und entsperrt. Der Deskriptor ist anschließend nicht mehr gültig.

Mögliche Laufzeitfehler:

Deskriptor des Validierungsverzeichnisses bzw. Validierungselementes

**ErrHdlInvalid**

(obj) ist ungültig.



## Kontakt



obj -> VldDelete(alpha1) : int

Validierungselement löschen

obj       Validierungsverzeichnis

alpha1   Objektname

      Löschresultat

ErrOk

      Löschen erfolgreich


ErrVldNameInvalid Elementname

      (alpha1) ist ungültig

ErrVldNoFile

      Validierungselement

Resultat int

      (alpha1) existiert 

      nicht

ErrVldLocked

      Validierungselement

      (alpha1) ist gesperrt

rDeadlock

      Verklemmung

      aufgetreten

Siehe   Verwandte Befehle, VldOpen()

Mit dieser Funktion wird das Validierungselement (alpha1) gelöscht. In (obj) wird der Deskriptor des Validierungsverzeichnisses angegeben.

### Resultat

Der Rückgabewert gibt Rückschlüsse über den Erfolg der Löschoperation. Folgende Werte können zurückgegeben werden:

<u>ErrOk</u>	Löschen erfolgreich
<u>ErrVldNameInvalid</u>	Elementname (alpha1) ist ungültig
<u>ErrVldNoFile</u>	Validierungselement (alpha1) existiert nicht
<u>ErrVldLocked</u>	Validierungselement (alpha1) ist gesperrt
<u>rDeadlock</u>	Verklemmung aufgetreten

### Beispiel

```
// Objekt 'Frm_Main' im Verzeichnis tVldDir löscht tVldDir->VldDelete('Frm_Main');
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor des Validierungsverzeichnisses (obj) ist ungültig.

## Kontakt



**VldDirOpen(int1) : handle**

**Validierungsverzeichnis öffnen**

**Verzeichnistyp**

**\_VldTypeDialog**

Verzeichnis für  
Validierungselemente

für Dialoge.

**\_VldTypeMenu**

Verzeichnis für  
Validierungselemente  
für Menüs.

**int1**

**\_VldTypeProcedure** Verzeichnis für

Validierungselemente  
für Prozeduren.

**\_VldTypeTable**

Verzeichnis für  
Validierungselemente  
für  
Datenbank-Tabellen.

Verzeichnis-Deskriptor oder

**Resultat handle \_ErrRights**

Keine  
ausreichende  
Berechtigung.



Siehe

**Verwandte Befehle, VldClose(),**

**Validierungselemente**

Mit dieser Funktion wird ein Verzeichnis von Validierungselementen geöffnet oder.

Hierfür muss in (int1) einer der folgenden Verzeichnistypen angegeben werden:

**\_VldTypeDialog**

Verzeichnis für Validierungselemente für Dialoge.

**\_VldTypeMenu**

Verzeichnis für Validierungselemente für Menüs.

**\_VldTypeProcedure**

Verzeichnis für Validierungselemente für Prozeduren.

**\_VldTypeTable**

Verzeichnis für Validierungselemente für Datenbank-Tabellen.

**Resultat**

Von dem Befehl wird der Deskriptor auf das Validierungsverzeichnis oder ein

Fehlercode zurückgegeben. Im Fall unzureichender Berechtigungen wird der

Fehlercode **\_ErrRights** zurückgegeben.

**Beispiel**

```
// Validierungsverzeichnis für Dialoge öffnetVldDir # VldDirOpen(_VldTypeDialog);if (tVldDir > 0
```

**Mögliche Laufzeitfehler:**

**\_ErrValueInvalid** Ungültiger Verzeichnistyp (int1) angegeben.

obj -> VldDirRead(int1[, alpha2]) : alpha  Inhalt eines

Validierungsverzeichnisses lesen

obj      Validierungsverzeichnis

Lesemodus

0            Das angegebene  
              Validierungselement  
              lesen

\_VldFirst Erstes  
              Validierungselement

lesen

\_VldLast Letztes  
int1        Validierungselement

lesen

\_VldNext Validierungselement

nach

Referenzeintrag

lesen

\_VldPrev Validierungselement

vor Referenzeintrag

lesen

alpha2    Referenzeintrag (optional)

Resultat alpha      Eintragsname

Verwandte Befehle,

Siehe

VldDirOpen()

Diese Funktion liest den Namen eines Validierungselementes aus dem Validierungsverzeichnis (obj).

Das Verzeichnis (obj) muss mit VldDirOpen() geöffnet worden sein.

Das Lesen der Verzeichniseinträge kann über folgende Optionen (int1) erfolgen:

- 0

Der Validierungselement mit dem in (alpha2) angegebenem Namen wird gelesen. Ist kein Element mit dem Namen vorhanden, wird der Eintrag mit dem nächst höheren Namen gelesen. Ist kein nächst höherer vorhanden, wird ein Leerstring zurückgegeben.

- \_VldFirst

Das erste Validierungselement wird gelesen.

- \_VldLast

Das letzte Validierungselement wird gelesen.

- \_VldNext

Das Validierungselement nach dem Referenzeintrag (alpha2) wird gelesen.

- \_VldPrev

Das Validierungselement vor dem Referenzeintrag (alpha2) wird gelesen.

## Kontakt

Konnte kein Eintrag gelesen werden (zum Beispiel, weil bei VldNext kein Folgeeintrag existiert), wird als Ergebnis eine leere Zeichenkette zurückgegeben ('').

Beispiel:

```
// Verzeichnis der Dialoge öffnetVldDir # VldDirOpen(_VldTypeDialog);if (tVldDir > 0){ // Erste
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor des Validierungsverzeichnisses (obj) ist ungültig.

## Kontakt

obj -> VldOpen(alpha1[, int2]) : handle 

Validierungselement öffnen bzw. erstellen

obj Validierungsverzeichnis

alpha1 Elementname

Optionen (optional)

\_VldLock Element für andere Benutzer

int2 sperren

\_VldCreate Element erstellen

Öffnungs-/Anlegeresultat

Deskriptor des  
Validierungselementes

oder

\_ErrVldNameInvalid Elementname

(alpha1) ungültig

\_ErrVldNoFile Validierungselement

(alpha1) ist nicht

vorhanden

\_ErrVldLocked Validierungselement

(alpha1) ist gesperrt

Resultat handle

\_ErrVldExists In einem zweiten

Client wurde ein

Validierungselement

mit dem gleichen

Namen angelegt

und noch nicht mit

VldClose()

geschlossen

Verklemmung

\_rDeadlock

aufgetreten

Siehe Verwandte Befehle, VldClose(),

Validierungselemente

Mit dieser Funktion wird ein Validierungselement geöffnet oder neu angelegt. In (obj) wird der Deskriptor des Validierungsverzeichnisses angegeben.

Die maximale Länge eines Validierungselementes beträgt 60 Zeichen. Der Name darf keine Steuerzeichen oder die Zeichen \* und ? enthalten.

Folgende Optionen (int2) können angegeben werden:

- VldCreate

Das Validierungselement wird im Validierungsverzeichnis erstellt.

- VldLock

Das Validierungselement wird beim Öffnen oder Anlegen für andere Benutzer gesperrt.



## Kontakt

Wird keine Sperroption angegeben, wird das Validierungselement mit einer gemeinsamen Sperre geöffnet.

Die Sperrung eines Validierungselementes bleibt bis zum Schließen des Objektes mit VldClose() oder bis sich der Benutzer von der Datenbank abmeldet erhalten. Änderungen an einem Validierungselement können nur bei einer exklusiven Sperre (siehe VldLock) vorgenommen werden.

Beispiele:

```
// Objekt 'Frm_Main' im Verzeichnis tVldDir öffnetVldElm # tVldDir->VldOpen('Frm_Main');// Objek
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor des Validierungsverzeichnisses (obj) ist ungültig.

## Kontakt

obj -> VldUpdate() : int        Änderungen an \_\_\_\_\_

Validierungselement übernehmen

obj \_\_\_\_\_ Deskriptor des Validierungselementes

Übernahmeresultat

ErrOk

Übernehmen erfolgreich

Resultat int ErrVldNoLock Validierungselement (obj) ist



nicht gesperrt

rDeadlock

Verklemmung aufgetreten

Siehe Verwandte Befehle

Mit diesem Befehl werden Änderungen an den Eigenschaften eines

Validierungselementes in der Datenbank gespeichert. Das Validierungselement muss dazu exklusiv gesperrt sein (siehe VldLock).

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor des Validierungselement (obj) ist ungültig.

Konstanten für Validierungsbefehle

Konstanten für Validierungsbefehle

Siehe Validierungsbefehle

- VldCreate
- VldFirst
- VldLast
- VldLock
- VldNext
- VldPrev
- VldStateDeleted
- VldStateIrrelevant
- VldStateModified
- VldStateNotVerified
- VldStateUndefined
- VldStateVerified
- VldTypeDialog
- VldTypeMenu
- VldTypeProcedure
- VldTypeTable



**\_VldCreate**

**Validierungselement erstellen**

Wert 4.096 /

**0x00001000**

**Verwandte**

Siehe **Befehle**,

**VldOpen()**

Option bei **VldOpen()** durch die ein neues **Validierungselement** erstellt werden kann.

**\_VldFirst**

Erstes Validierungselement lesen

Wert <sup>1</sup>/

**0x00000001**

**Verwandte**

Siehe **Befehle**,

**VldDirRead()**

Option bei **VldDirRead()** durch die das erste Validierungselement in einem Validierungsverzeichnis gelesen werden kann.

**\_VldLast**

**Letztes Validierungselement lesen**

Wert <sup>2/</sup>

**0x00000002**

**Verwandte**

**Siehe Befehle,**

**VldDirRead()**

**Option bei VldDirRead() durch die das letzte Validierungselement in einem Validierungsverzeichnis gelesen werden kann.**

## Kontakt

**\_VldLock**

**Validierungselement für andere Benutzer sperren**

Wert <sup>8</sup>/

**0x00000008**

**Verwandte**

Siehe **Befehle**,

**VldOpen()**

Option bei **VldOpen()** durch die ein **Validierungselement** beim Öffnen/Anlegen für andere Benutzer gesperrt werden kann.

## Kontakt

**\_VldNext**

**Element nach Referenzeintrag lesen**

Wert <sup>4</sup>/

**0x00000004**

**Verwandte**

Siehe **Befehle**,

**VldDirRead()**

Option bei **VldDirRead()** durch die das **Validierungselement** nach dem Referenzeintrag in einem Validierungsverzeichnis gelesen werden kann.

**\_VldPrev**

**Element vor Referenzeintrag lesen**

Wert<sup>3/</sup>

**0x00000003**

**Verwandte**

Siehe **Befehle**,

**VldDirRead()**

Option bei **VldDirRead()** durch die das **Validierungselement** vor dem Referenzeintrag in einem Validierungsverzeichnis gelesen werden kann.

## Kontakt

\_VldStateDeleted

Das Element wurde gelöscht

Wert 6

Siehe Flags,

Validierungselemente

Wird die Eigenschaft Flags eines Validierungselementes auf den Wert

\_VldStateDeleted gesetzt, gilt das Referenzelement des Validierungselementes als gelöscht.



Die Eigenschaft Flags hat bei Validierungselementen einen rein informativen Charakter. Die Konstanten \_VldState... stehen hierbei für Status, die vom Validierungs-Editor interpretiert werden. Bei der Verwendung der Validierungsbefehle können auch eigene Werte verwendet werden.

\_VldStateIrrelevant

Das Element ist nicht für die Validierung relevant Wert 2

**Siehe** Flags,  
Validierungselemente

Wird die Eigenschaft Flags eines Validierungselementes auf den Wert

\_VldStateIrrelevant gesetzt, gilt das Referenzelement des Validierungselementes als nicht relevant für die Validierung.



Die Eigenschaft Flags hat bei Validierungselementen einen rein informativen Charakter. Die Konstanten \_VldState... stehen hierbei für Status, die vom Validierungs-Editor interpretiert werden. Bei der Verwendung der Validierungsbefehle können auch eigene Werte verwendet werden.



## Kontakt

**\_VldStateModified**

Das Element wurde modifiziert

Wert 5

**Siehe** Flags,

Validierungselemente

Wird die Eigenschaft Flags eines Validierungselementes auf den Wert

\_VldStateModified gesetzt, gilt das Referenzelement des Validierungselementes als modifiziert.



Die Eigenschaft Flags hat bei Validierungselementen einen rein informativen Charakter. Die Konstanten \_VldState... stehen hierbei für Status, die vom Validierungs-Editor interpretiert werden. Bei der Verwendung der Validierungsbefehle können auch eigene Werte verwendet werden.

**\_VldStateNotVerified**

Das Element wurde noch nicht validiert

Wert 3

Siehe Flags,

Validierungselemente

Wird die Eigenschaft Flags eines Validierungselementes auf den Wert

\_VldStateNotVerified gesetzt, gilt das Referenzelement des Validierungselementes als noch nicht validiert.



Die Eigenschaft Flags hat bei Validierungselementen einen rein informativen Charakter. Die Konstanten \_VldState... stehen hierbei für Status, die vom Validierungs-Editor interpretiert werden. Bei der Verwendung der Validierungsbefehle können auch eigene Werte verwendet werden.

**\_VldStateUndefined**

Validierungszustand nicht definiert

Wert 1

**Siehe** Flags,

Validierungselemente

Wird die Eigenschaft Flags eines Validierungselementes auf den Wert

\_VldStateUndefined gesetzt, gilt der Zustand des Elementes als nicht definiert.



Die Eigenschaft Flags hat bei Validierungselementen einen rein informativen Charakter. Die Konstanten \_VldState... stehen hierbei für Status, die vom Validierungs-Editor interpretiert werden. Bei der Verwendung der Validierungsbefehle können auch eigene Werte verwendet werden.

## Kontakt

**\_VldStateVerified**

Das Element wurde validiert

Wert 4

**Siehe** Flags,

Validierungselemente

Wird die Eigenschaft Flags eines Validierungselementes auf den Wert

\_VldStateVerified gesetzt, gilt das Referenzelement des Validierungselementes als validiert.



Die Eigenschaft Flags hat bei Validierungselementen einen rein informativen Charakter. Die Konstanten \_VldState... stehen hierbei für Status, die vom Validierungs-Editor interpretiert werden. Bei der Verwendung der Validierungsbefehle können auch eigene Werte verwendet werden.

**\_VldTypeDialog**

**Validierungsverzeichnis für Dialoge**

**Wert 14**

**Verwandte**

Siehe **Befehle**,

**VldDirOpen()**

Option bei **VldDirOpen()**, mit der das Validierungsverzeichnis der Dialoge geöffnet werden kann.

**\_VldTypeMenu**

**Validierungsverzeichnis für Menüs**

**Wert 15**

**Verwandte**

Siehe **Befehle**,

**VldDirOpen()**

Option bei **VldDirOpen()**, mit der das Validierungsverzeichnis der Menüs geöffnet werden kann.

**\_VldTypeProcedure**

**Validierungsverzeichnis für Prozeduren**

**Wert 22**

**Verwandte**

**Siehe Befehle,**

**VldDirOpen()**

**Option bei VldDirOpen(), mit der das Validierungsverzeichnis der Prozeduren geöffnet werden kann.**

**\_VldTypeTable**

**Validierungsverzeichnis für Tabellen**

**Wert 13**

**Verwandte**

Siehe **Befehle**,

**VldDirOpen()**

Option bei **VldDirOpen()**, mit der das Validierungsverzeichnis der Tabellen geöffnet werden kann.



Funktionen der Systemumgebung

Funktionen der Systemumgebung

Siehe **Befehlsgruppen** **Befehlsliste**,

- **Befehle für Systemobjekte**
- **Dateibefehle**
- **Netzwerkinformationsbefehle**
- **Systemfunktionen**

## Kontakt

### Dateibefehle (Extern)

Befehle zum Umgang mit externen Dateien

#### Befehlsgruppen,

Siehe Befehlsliste,

#### Beispiel

Mit den folgenden Befehlen können externe Dateien und das Massenspeichersystem verarbeitet werden. Die meisten Anweisung geben einen Fehlerwert (\_ErrFsi...) zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt und ausgewertet werden.

### Befehle

- FsiAttributes
- FsiClose
- FsiDate
- FsiDelete
- FsiDirClose
- FsiDirOpen
- FsiDirRead
- FsiDiskInfo
- FsiFileCompress
- FsiFileInfo
- FsiFileProcess
- FsiFileUncompress
- FsiLock
- FsiMark
- FsiMonitorAdd
- FsiMonitorClose
- FsiMonitorControl
- FsiMonitorOpen
- FsiMonitorRemove
- FsiOpen
- FsiPath
- FsiPathChange
- FsiPathCreate
- FsiPathDelete
- FsiRead
- FsiReadMem
- FsiRename
- FsiSeek
- FsiSeek64
- FsiSize
- FsiSize64
- FsiSplitName
- FsiStamp
- FsiTime
- FsiWrite
- FsiWriteMem

### Konstanten

- ComprFmtDeflate
- ComprFmtGzip
- ComprFmtZlib
- ComprLvlDefault
- FsiAcsR
- FsiAcsRW
- FsiAcsW
- FsiANSI
- FsiAppend
- FsiAttrArchive
- FsiAttrDir
- FsiAttrExec
- FsiAttrHidden
- FsiAttrRead
- FsiAttrSystem
- FsiAttrWrite
- FsiBuffer
- FsiCompressFast
- FsiCompressMed
- FsiCompressSlow
- FsiCompressStd
- FsiCreate
- FsiCreateNew
- FsiDecode
- FsiDeleteOnClose
- FsiDenyNone
- FsiDenyR
- FsiDenyRW
- FsiDenyW
- FsiDiskAvailMB
- FsiDiskExists
- FsiDiskFree
- FsiDiskFreeMB
- FsiDiskReady
- FsiDiskTotal
- FsiDiskTotalMB
- FsiDtAccessed
- FsiDtCreated
- FsiDtModified
- FsiEncrypt
- FsiFileCRC32
- FsiFileMD5
- FsiFileRMD160
- FsiFileSHA1
- FsiFileSHA256
- FsiFileSHA384
- FsiFileSHA512
- FsiFileVersion
- FsiFileVersionHex
- FsiGroupR
- FsiGroupW

- FsiMonActionCreate
- FsiMonActionDelete
- FsiMonActionModify
- FsiMonActionRename
- FsiMonFlagsSubDirs
- FsiMonitorStart
- FsiMonitorStop
- FsiNameC16
- FsiNameE
- FsiNameN
- FsiNameNE
- FsiNameP
- FsiNamePN
- FsiNamePNE
- FsiNamePP
- FsiNameUtf8
- FsiNoCache
- FsiOtherR
- FsiOtherW
- FsiPure
- FsiStdRead
- FsiStdWrite
- FsiSyncWrite
- FsiTruncate
- FsiUserR
- FsiUserW

## Kontakt



**FsiAttributes(alpha1[, int2]) : int**

**Dateiattribute ermitteln/setzen**

**alpha1**    **Dateiname/-pfad**

**Neue Dateiattribute (optional)**

**\_FsiAttrHidden**    **Versteckte Datei**

**ermitteln/setzen**

**\_FsiAttrSystem**    **Systemdatei ermitteln/setzen**

**\_FsiAttrDir**    **Verzeichnis ermitteln**

**\_FsiAttrArchive**    **Archivdatei ermitteln/setzen**

**\_FsiAttrRead**    **Leseberechtigung ermitteln**

**int2**    **\_FsiAttrWrite**    **Schreibberechtigung**

**ermitteln/setzen**

**\_FsiAttrExec**    **Ausführbarkeit ermitteln (nur**

**UNIX)**

**\_FsiNameC16**    **Dateiname ist im CONZEPT**

**16-Zeichensatz angegeben**

**\_FsiNameUtf8**    **Dateiname ist im**

**UTF-8-Zeichensatz**

**angegeben**

**Ermittlungs-/Setzungsergebnis**

**Aktuelle Dateiattribute oder**

**\_ErrFsiNoFile**    **Dateiname/-pfad**  
**(alpha1) nicht**

**vorhanden**

**\_ErrFsiAccessDenied**    **Berechtigungen**  
**nicht**

**ausreichend**

**Resultat**    **int**    **\_ErrFsiDriveInvalid**    **Auf Laufwerk** 

**(alpha1) kann**

**nicht**

**zugegriffen**

**werden**

**\_ErrFsiSharingViolation**    **Zugriffskonflikt**

**aufgetreten**

**\_ErrFsiLockViolation**    **Sperrkonflikt**

**aufgetreten**

**Siehe**    **Verwandte Befehle**

**Mit dieser Funktion können die Attribute einer Datei ermittelt und gesetzt werden.**

**Das Resultat enthält die aktuellen Attribute der Datei (Resultat >= 0) oder einen Fehlerwert. Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.**

**Beim Verändern von Attributen sollten vorher unbedingt die aktuellen Attribute der Datei gelesen werden und diese dann modifiziert werden.**

**Beispiele:**

## Kontakt

```
// Überprüfen, ob eine Datei schreibgeschützt ist if ((FsiAttributes(_Sys->spPathMyDocuments + '\T  
// Setzen des Archiv-Attributes einer Datei mit UTF-8-Zeichen im NamentAttr # FsiAttributes(tFile
```

obj ->



**FsiClose()**

Datei schließen

obj    Datei-Deskriptor

Verwandte

Siehe Befehle,

FsiOpen()

Mit dieser Funktion wird eine mittels FsiOpen() geöffnete externe Datei wieder geschlossen. In (obj) wird der Deskriptor der offenen Datei übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Datei-Deskriptor (obj) ungültig

obj -> FsiDate(int1[, date2]) : date  Dateiinformationen

ermitteln/setzen (Datumswerte)

obj      Datei-Deskriptor

Informationstyp

FsiDtModified Letztes

Änderungsdatum

ermitteln/setzen

int1      FsiDtAccessed Letztes

Zugriffsdatum

ermitteln/setzen

FsiDtCreated Erstellungsdatum

ermitteln/setzen

date2      Neues Datum (optional)

Resultat date      Aktuelles Datum

Verwandte Befehle, FsiOpen(),

Siehe

FsiTime(), FsiStamp()

Mit dieser Funktion können Datumswerte der externen Datei (obj) abgefragt (zwei Argumente) oder geändert werden (drei Argumente).

Bei FAT-Dateisystemen ist nur das Datum der letzten Änderung verfügbar. Bei Linux-Dateisystemen ist das Datum des letzten Zugriffs nicht verfügbar. Bei NTFS-Dateisystemen sind alle Datumswerte verfügbar. Sofern ein Datumswert nicht verfügbar ist, wird ein leeres Datum zurückgeliefert.

FsiDate() kann auch zur Abfrage der Datumswerte eines Verzeichniseintrags benutzt werden (siehe FsiDirRead()). Dazu wird in (obj) der Deskriptor des Verzeichnisses übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Datei-Deskriptor (obj) ungültig



## Kontakt

**FsiDelete(alpha1[, int2]) :**



**int**

**Datei löschen**

**alpha1**     **Dateiname/-pfad**

**Optionen (optional)**

**int2**     **FsiNameC16** **Dateiname/-pfad (alpha1) wird im  
CONZEPT 16-Zeichensatz  
erwartet (Standard)**

**FsiNameUtf8** **Dateiname/-pfad (alpha1) wird  
im UTF-8-Zeichensatz  
erwartet**

**Löschergebnis**

**ErrOk**

**Löschen**

**erfolgreich**

**ErrFsiNoFile**

**Dateiname/-pfad  
(alpha1) nicht**

**vorhanden**

**ErrFsiAccessDenied**

**Berechtigungen  
nicht**

**Resultat int**

**ausreichend**



**ErrFsiDriveInvalid**

**Auf Laufwerk  
(alpha1) kann  
nicht**

**zugegriffen**

**werden**

**ErrFsiSharingViolation** **Zugriffskonflikt**

**aufgetreten**

**ErrFsiLockViolation**

**Datei gesperrt**

**Siehe**     **Verwandte Befehle**, **FsiPathDelete()**

**Diese Funktion löscht die externe Datei mit dem Namen (alpha1). Verzeichnisse können mit FsiPathDelete() gelöscht werden. Die Anweisung gibt einen Fehlerwert zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.**

**Wird im optionalen Argument (int2) FsiNameUtf8 angegeben, wird der**

**Dateiname/-pfad (alpha1) als UTF-8-Zeichenkette erwartet. Somit ist es auch möglich  
Dateien mit Umlauten anderer Sprachen zu löschen.**

**Mögliche Laufzeitfehler:**

**ErrValueInvalid** **Es wurde eine ungültige Option (int2) angegeben.**

obj -> FsiDirClose()

Verzeichnis schließen

obj Verzeichnis-Deskriptor

Verwandte Befehle,

**Siehe** FsiDirOpen()

Mit dieser Funktion kann ein mit FsiDirOpen() geöffnetes Verzeichnis wieder geschlossen werden. In (obj) wird der Deskriptor des Verzeichnisses übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Verzeichnis-Deskriptor (obj) ungültig

**FsiDirOpen(alpha1, int2) :**



**handle**

**Verzeichnis öffnen**

**alpha1**    **Dateiname/-pfad**

**Optionen**

**\_FsiAttrHidden**    **Versteckte Dateien**

**ermitteln**

**\_FsiAttrSystem**    **Systemdateien ermitteln**

**\_FsiAttrDir**    **Verzeichnisse ermitteln**

**int2**    **\_FsiNameC16**    **Dateiname ist im  
CONZEPT 16-Zeichensatz**

**angegeben**

**\_FsiNameUtf8**    **Dateiname ist im  
UTF-8-Zeichensatz**

**angegeben**

**Resultat** **handle**    **Verzeichnis-Deskriptor**



**Verwandte Befehle, FsiDirClose(),**

**Siehe**    **FsiDirRead(), FsiAttributes(),  
Fehlerwerte**

Mit dieser Funktion wird ein Verzeichnis zum Lesen geöffnet. Anschließend können mit dem Befehl **FsiDirRead()** die Verzeichniseinträge gelesen werden. Ist der Dateiname/-pfad (alpha1) leer, können alle Einträge des aktuellen Verzeichnisses gelesen werden. Beim Lesen der Verzeichniseinträge ist es ebenfalls möglich in (alpha1) auch gleich eine Dateimaske zur Filterung der Einträge mit anzugeben.



Existiert das Verzeichnis nicht, wird trotzdem ein gültiger Deskriptor zurückgegeben. Die Existenz eines Verzeichnisses kann mit der Funktion **FsiAttributes()** geprüft werden.

**Folgende Optionen (int2) sind zulässig:**

- **\_FsiAttrHidden**

**Versteckte Dateien werden gelesen**

- **\_FsiAttrSystem**    **Systemdateien**

**werden gelesen**

- **\_FsiAttrDir**

**Verzeichnisse werden gelesen**

- **\_FsiNameC16**

**Dateiname ist im CONZEPT 16-Zeichensatz angegeben**

- **\_FsiNameUtf8**

**Dateiname ist im UTF-8-Zeichensatz angegeben**

**Beispiel:**

```
// Lesen aller .dat-Dateien im aktuellen VerzeichnistDirHdl # FsiDirOpen('*', _FsiAttrHidden)
```

## Kontakt

```
// Liste aller Dateien eines Verzeichnisses erstellensub WriteFileList( aPath : alpha(4096);)
```

obj -> FsiDirRead() :



alpha

Verzeichniseintrag lesen

obj Verzeichnis-Deskriptor

Resultat alpha Name des

Verzeichniseintrags

Verwandte Befehle,

Siehe

FsiDirOpen()

Mit dieser Funktion können die Einträge eines mit FsiDirOpen() geöffneten Verzeichnisses gelesen werden. In (obj) wird der Deskriptor des Verzeichnisses übergeben. Das Resultat ist der Name der nächsten Datei. Wenn das Resultat ein leerer Alphawert ist, sind keine weiteren Einträge im Verzeichnis vorhanden.

Ist bei FsiDirOpen() die Option \_FsiNameUtf8 angegeben, werden die gelesenen Namen im UTF-8-Zeichensatz zurückgegeben.

Datum, Uhrzeit und Größe der Datei können mit FsiDate(), FsiTime(), FsiStamp() und FsiSize() ermittelt werden. Die Dateiattribute können mit FsiAttributes() abgefragt werden.

Mögliche Laufzeitfehler:

\_ErrHdlInvalid Verzeichnis-Deskriptor (obj) ungültig

## Kontakt



**FsiDiskInfo(alpha1, int2) : int**

**Datenträgerinformationen ermitteln**

**alpha1** Laufwerksbuchstabe

**Dateiattribute**

**\_FsiDiskFree** Freie Kapazität in KB

ermitteln

**\_FsiDiskFreeMB** Freie Kapazität in MB

ermitteln

**\_FsiDiskTotal** Gesamte Kapazität in

KB ermitteln

**int2** **\_FsiDiskTotalMB** Gesamte Kapazität in

MB ermitteln

**\_FsiDiskAvailMB** Verfügbare Kapazität in

MB ermitteln

**\_FsiDiskReady** Datenträgerbereitschaft

ermitteln

**\_FsiDiskExists** Datenträgerexistenz

ermitteln

**Resultat int** Datenträgerinformation

**Siehe** Verwandte Befehle

Mit dieser Funktion können Informationen zu einem Datenträger ermittelt werden. In (alpha1) wird der Buchstabe des Laufwerks übergeben.

Folgende Optionen (int2) sind zulässig:

- **\_FsiDiskFree**

Das Resultat ist die freie Laufwerkskapazität in Kilobyte. Kann die Information nicht ermittelt werden, wird 0 zurückgegeben. Sind mehr als 2 Terabyte frei, wird **\_MaxInt** zurückgegeben. In diesem Fall sollte die Option **\_FsiDiskFreeMB** verwendet werden.

- **\_FsiDiskFreeMB**

Das Resultat ist die freie Laufwerkskapazität in Megabyte. Kann die Information nicht ermittelt werden, wird 0 zurückgegeben. Sind mehr als 2 Petabyte frei, wird **\_MaxInt** zurückgegeben.

- **\_FsiDiskTotal**

Das Resultat ist die gesamte Laufwerkskapazität in Kilobyte. Kann die Information nicht ermittelt werden, wird 0 zurückgegeben. Sind mehr als 2 Terabyte vorhanden, wird **\_MaxInt** zurückgegeben. In diesem Fall sollte die Option **\_FsiDiskTotalMB** verwendet werden.

- **\_FsiDiskTotalMB**

Das Resultat ist die gesamte Laufwerkskapazität in Megabyte. Kann die Information nicht ermittelt werden, wird 0 zurückgegeben. Sind mehr als 2 Petabyte vorhanden, wird **\_MaxInt** zurückgegeben.

- **\_FsiDiskAvailMB**

## Kontakt

Das Resultat ist die verfügbare Laufwerkskapazität in Megabyte. Kann die Information nicht ermittelt werden, wird 0 zurückgegeben. Sind mehr als 2 Petabyte verfügbar, wird MaxInt zurückgegeben. Die verfügbare Laufwerkskapazität wird durch Datenträgerkontingente festgelegt.

- FsiDiskReady

Das Resultat ist 1, wenn das Laufwerk bereit ist, ansonsten 0.

- FsiDiskExists

Das Resultat ist 1, wenn das Laufwerk existiert, ansonsten 0.



Der Befehl wird unter Linux nicht unterstützt. Als Resultat wird 0 zurückgegeben.

### Beispiele:

```
// Ermitteln des freien Speichers auf Laufwerk C:tDiskFree # FsiDiskInfo('C', _FsiDiskFree);// Er
```

**FsiFileCompress(alpha1, int2[,  
int3[, int4[, int5[, alpha6]]]]) :**  
**int**



**Externe Dateien komprimieren**

**alpha1**    Name der Quelldatei

**Kompressionsformat**

**\_ComprFmtDeflate**    DEFLATE-Format

**int2**        **\_ComprFmtGzip**        GZIP-Format

**\_ComprFmtZlib**        ZLIB-Format

**int3**        Kompressionsstufe (optional)

**int4**        Quellposition (optional)

**int5**        Quelllänge (optional)

**alpha6**    Name der Zieldatei (optional)

**Resultat int**    Fehlerwert



**Verwandte Befehle,**

**FsiFileUncompress(),**

Siehe

**MemCompress(), FsiFileProcess(),**

**Fehlerwerte**

Mit dieser Anweisung können externe Dateien komprimiert werden. Die Quelldatei wird in (alpha1) angegeben.

Es muss eines der folgenden Kompressionsformate (int2) angegeben werden:

**\_ComprFmtDeflate**    DEFLATE-Format

**\_ComprFmtGzip**        GZIP-Format

**\_ComprFmtZlib**        ZLIB-Format

Als Kompressionsstufe (int3) können Werte zwischen 0 (keine Komprimierung) und 9 (maximale Komprimierung) angegeben werden. Alternativ wird mit **\_ComprLvlDefault** die Standard-Komprimierungsstufe angegeben.

Im Parameter (int4) kann die Quellposition angegeben werden. Ist dieser Wert nicht angegeben oder 0, werden die Daten ab Beginn der Quelldatei komprimiert.

Der Parameter (int5) gibt die zu komprimierende Länge an. Ist dieser Wert nicht angegeben oder 0 wird der restliche Inhalt (nach der Quellposition) der Datei komprimiert.

Optional kann im Parameter (alpha6) eine Zieldatei angegeben werden. Ist diese bereits vorhanden, wird sie überschrieben. Wurde keine Zieldatei angegeben, oder ist sie mit der Quelldatei identisch, wird die Quelldatei überschrieben.

**Beispiele**

```
// Inhalt der Datei 'Test.txt' in neuer Datei im GZIP-Format komprimierenFsiFileCompress('Test.tx
```

**Fehlerwerte**

Zusätzlich zu den Fehlerwerten für externe Dateioperationen (**\_ErrFsi...**) können folgende Fehlerwerte von der Funktion zurückgegeben werden:



## Kontakt

ErrOk      Kein Fehler aufgetreten.

ErrGeneric Interner Fehler aufgetreten.

Mögliche Laufzeitfehler:

ErrMemExhausted Nicht genug Speicher vorhanden.

ErrValueInvalid      Im Kompressionsformat (int2) oder Kompressionsstufe (int3) wurde ein ungültiger Wert angegeben.

## Kontakt



**FsiFileInfo(alpha1, int2) : alpha**

Informationen zu einer externen Datei ermitteln  
Pfad und Dateiname der

**alpha1** externen Datei

**int2** zu ermittelnde Information

Resultat alpha gewünschte Information  —

Siehe Verwandte Befehle

Mit dieser Anweisung können Informationen zu einer externen Datei ermittelt werden. Der Pfad und der Name der Datei wird in (alpha1) übergeben. (int2) bestimmt die zu ermittelnde Information. Folgende symbolische Konstanten können in (int2) übergeben werden:

<u>FsiFileCRC32</u>	CRC32-Prüfsumme der Datei
<u>FsiFileMD5</u>	MD-5 Hash-Wert der Datei
<u>FsiFileRMD160</u>	RIPEMD-160-Standard Hash-Wert der Datei
<u>FsiFileSHA1</u>	SHA-1 Hash-Wert der Datei
<u>FsiFileSHA256</u>	SHA-256 Hash-Wert der Datei
<u>FsiFileSHA384</u>	SHA-384 Hash-Wert der Datei
<u>FsiFileSHA512</u>	SHA-512 Hash-Wert der Datei
<u>FsiFileVersion</u>	Version der Datei
<u>FsiFileVersionHex</u>	Version in Hexadezimalziffern der Datei



Die Verfahren FsiFileCRC32, FsiFileMD5, FsiFileRMD160 und FsiFileSHA1 sind nicht kollisionsicher und sollten daher nicht zur Integritätsprüfung von Dateien verwendet werden.



Es ist zu beachten, dass die Ermittlung des Hash-Wertes bei großen Dateien eine gewisse Zeit beanspruchen kann.

Kann die Datei in (alpha1) nicht geöffnet werden, gibt der Befehl einen Leerstring zurück.

Beispiel:

```
if (ART.aBildSHA256 != FsiFileInfo(tPicPath + ART.aBild, _FsiFileSHA256)){ // Datei wurde veränd
```

Auftretende Fehler setzen den globalen Fehlerwert und können mit ErrGet() abgefragt werden. Neben den Fehlern für Externe Dateioperationen können folgende Fehler gesetzt werden:

- ErrData - Die Datei existiert, enthält aber keine Versionsinformationen.



**FsiFileProcess(alpha1, alpha2, int3[, alpha4]) : int**

**Externe Dateien komprimieren und verschlüsseln sowie entschlüsseln und dekomprimieren**

**alpha1** Name der zu verarbeitenden Datei

**alpha2** Name der Zieldatei

**Verarbeitungsoptionen**

**FsiEncrypt** Datei verschlüsseln

**FsiCompress...** Datei komprimieren

**FsiDecode** Datei dekomprimieren  
und entschlüsseln

**FsiFileMD5** MD5-Hash-Wert an die

**FsiFileCRC32** Datei anhängen  
CRC32-Prüfsumme an  
die Datei anhängen

**int3** **FsiFileRMD160** RIPEMD-160-Hash-Wert

**FsiFileSHA1** an die Datei anhängen  
SHA-1-Hash-Wert an die

**FsiFileSHA256** Datei anhängen  
SHA-256-Hash-Wert an

**FsiFileSHA384** die Datei anhängen  
SHA-384-Hash-Wert an

**FsiFileSHA512** die Datei anhängen  
SHA-512-Hash-Wert an  
die Datei anhängen

**alpha4** zu verwendender Schlüssel (optional)

**Resultat** **int** Ergebnis der Dateiverarbeitung  **Verwandte**

**Befehle**, **FsiFileCompress()**, **FsiFileUncompress()**,  
**FsiOpen()**,

**Siehe** **FsiRead()**, **FsiWrite()**, **Fehlerwerte**,  
**Verschlüsselung und Komprimierung**  
**(Blog)**

Mit dieser Anweisung können externe Dateien komprimiert / verschlüsselt beziehungsweise entschlüsselt / dekomprimiert werden. Zusätzlich ist es mit **FsiFileProcess()** auch möglich die Dateien mit einem Hash-Wert zu versehen. Als Quelldatei wird die in (alpha1) angegebene externe Datei verwendet. Das Ergebnis der Verarbeitung wird in die Datei (alpha2) geschrieben. Wird der Parameter (alpha2) nicht angegeben, wird die Quelldatei durch die entstandene Datei ersetzt.

Eine Datei die mit dem Befehl **FsiFileProcess()** bearbeitet wurde, wird mit einem Dateifuß versehen. Dieser wird verwendet, um zu erkennen, ob eine Datei von CONZEPT 16 bereits komprimiert beziehungsweise verschlüsselt wurde und entsprechend wiederhergestellt werden kann. In diesem "Stempel" werden ebenfalls zusätzliche Informationen wie der ermittelte Hash-Wert der Datei abgelegt. Der Dateifuß wird beim Entschlüsseln beziehungsweise Dekomprimieren wieder entfernt.

Im Parameter (int3) werden die Optionen zur Dateiverarbeitung kombiniert. Folgende Optionen sind dabei möglich:

## Kontakt

- **FsiEncrypt**

Die Datei wird verschlüsselt.

- **FsiCompress...**

Die Datei wird komprimiert. Dabei können die Kompressionsstufen

FsiCompressFast, FsiCompressMed, FsiCompressStd und FsiCompressSlow verwendet werden.

- **FsiDecode**

Die Datei wird entschlüsselt und dekomprimiert.

- **FsiFileMD5**

Mit FsiFileMD5 (Message-Digest Algorithm 5) kann an eine Datei auch deren Hash-Wert angehängt werden. Wird zusätzlich zur Hash-Wert Funktionalität auch eine Komprimierung beziehungsweise Verschlüsselung durchgeführt, berechnet die Funktion den Hash-Wert erst nach der Dateiverarbeitung. Der Hash-Wert wird dann wieder bei der Entschlüsselung / Dekomprimierung der Datei mit dem Dateiinhalt verglichen. Mit dieser Funktion können Manipulationen an Dateien festgestellt werden. In diesem Fall wird ErrData zurückgegeben.

- **FsiFileCRC32**

Wie bei FsiFileMD5 wird an die Datei eine CRC32-Prüfsumme angehängt.

- **FsiFileRMD160**

Wie bei FsiFileMD5 wird an die Datei ein Hash-Wert angehängt. Dieser wird nach RIPEMD-160-Standard berechnet.

- **FsiFileSHA1**

Wie bei FsiFileMD5 wird an die Datei ein SHA-1-Hash-Wert angehängt.

- **FsiFileSHA256**

Wie bei FsiFileMD5 wird an die Datei ein SHA-256-Hash-Wert angehängt.

- **FsiFileSHA384**

Wie bei FsiFileMD5 wird an die Datei ein SHA-384-Hash-Wert angehängt.

- **FsiFileSHA512**

Wie bei FsiFileMD5 wird an die Datei ein SHA-512-Hash-Wert angehängt.

Für das Verschlüsseln beziehungsweise Entschlüsseln der Datei kann in (alpha4) optional ein eigener Schlüssel angegeben werden. Der Schlüssel darf eine maximale Länge von 64 Zeichen haben und muss beim Verschlüsselungs- und Entschlüsselungsvorgang identisch sein. Wird der Parameter (alpha4) nicht angegeben, wird ein interner Schlüssel verwendet.



Die Option FsiEncrypt sollte immer mit einer der FsiFile...-Optionen kombiniert werden.



## Kontakt

Die Verfahren FsiFileCRC32, FsiFileMD5, FsiFileRMD160 und FsiFileSHA1 sind nicht kollisionsicher und sollten daher nicht zur Integritätsprüfung von Dateien verwendet werden.



Bei den FsiCompress...-Optionen findet die Komprimierung in einem internen Format statt. Die Dateien können daher nur mit CONZEPT 16 wieder entpackt werden. Mit FsiFileCompress() können Dateien hingegen in einem standardisierten Format komprimiert werden.


Folgende Werte können von der Funktion zurückgegeben werden:

<u>ErrOk</u>	Dateiverarbeitung erfolgreich durchgeführt.
<u>ErrOutOfMemory</u>	Speicher konnte nicht angefordert werden
<u>ErrData</u>	Datenfehler in der externen Datei
<u>ErrDecryption</u>	Fehler bei der Dateientschlüsselung mit dem angegebenen Schlüssel
<u>ErrFsiReadFault</u>	Fehler beim Lesen der externen Datei
<u>ErrFsiWriteFault</u>	Fehler beim Schreiben der externen Datei

## Kontakt

**FsiFileUncompress(alpha1[, int2[, int3[, alpha4]]) : int** Externe Dateien dekomprimieren



alpha1      Name der  
            **Quelldatei**  
            **Quellposition**  
int2  
            **(optional)**  
            **Quelllänge**  
int3  
            **(optional)**  
            **Name der**  
**alpha4**      **(optional)**  
**Resultat int Fehlerwert**  —

**Zieldatei**

**Verwandte**  
**Befehle,**  
Siehe **FsiFileCompress(),**  
**MemUncompress(),**  
**FsiFileProcess(),**  
**Fehlerwerte**

Mit dieser Anweisung können externe Dateien dekomprimiert werden. Die Quelldatei wird in (alpha1) angegeben.

Im Parameter (int2) kann die Quellposition angegeben werden. Ist dieser Wert nicht angegeben oder 0, werden die Daten ab Beginn der Quelldatei komprimiert.

Der Parameter (int3) gibt die zu komprimierende Länge an. Ist dieser Wert nicht angegeben oder 0 wird der restliche Inhalt (nach der Quellposition) der Datei komprimiert.

Optional kann im Parameter (alpha4) eine Zieldatei angegeben werden. Ist diese bereits vorhanden, wird sie überschrieben. Wurde keine Zieldatei angegeben, oder ist sie mit der Quelldatei identisch, wird die Quelldatei überschrieben.

Zusätzlich kann eine Zielposition (int5) angegeben werden, wenn nicht an den Anfang der Zieldatei geschrieben werden soll.

### Beispiele

```
// Inhalt der Datei 'Test.txt.gz' in neuer Datei dekomprimierenFsiFileUncompress('Test.txt.gz', 0
```

### Fehlerwerte

Zusätzlich zu den Fehlerwerten für externe Dateioperationen (ErrFsi...) können folgende Fehlerwerte von der Funktion zurückgegeben werden:

Folgende Fehlerwerte können von der Funktion zurückgegeben werden:

ErrOk      Kein Fehler aufgetreten.  
ErrData    Komprimierte Daten sind inkonsistent oder Quelldatei (alpha1) ist leer.  
ErrGeneric Interner Fehler aufgetreten.

Mögliche Laufzeitfehler

ErrMemExhausted Nicht genug Speicher vorhanden.

## Kontakt

obj -> FsiLock(int1, int2, logic3) :  
int



Dateibereich sperren/entsperren

obj Datei-Deskriptor

int1 Sperrbereichsposition

int2 Sperrbereichsgröße

logic3 Sperren/Entsperren

Sperr-/Entsperrresultat

\_\_ErrOk

Sperren/Entsperren

erfolgreich

\_\_ErrFsiAccessDenied

Berechtigung nicht

Resultat int

ausreichend



\_\_ErrFsiSharingViolation Zugriffskonflikt

aufgetreten

\_\_ErrFsiLockViolation

Sperrkonflikt

aufgetreten

Siehe Verwandte Befehle

Mit dieser Funktion kann ein Bereich der externen Datei (obj) gesperrt (logic3 = true) oder entsperrt werden (logic3 = false). Der Bereich wird durch seine Position (int1) ab Dateianfang und seine Größe (int2) festgelegt. FsiLock() dient zur Synchronisation von Zugriffen auf gemeinsam benutzte externe Dateien. Lese- oder Schreibzugriffe auf gesperrte Bereiche sind dabei von anderen Benutzern nicht möglich.

Bei der Synchronisation von Zugriffen sollte vor einer Lese- bzw. Schreiboperation der entsprechende Bereich gesperrt werden.

Die Anweisung gibt einen Fehlerwert (\_\_ErrFsi...) zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.

Beispiel:

```
// Dateibereichssperrung erfolgreich if (tHandle->FsiLock(tPos, 100, true) == __ErrOk){ // Dateizei
```

Mögliche Laufzeitfehler:

\_\_ErrHdlInvalid Datei-Deskriptor (obj) ungültig



## Kontakt

obj -> FsiMark([int1]) : byte 

Dateitrennzeichen ermitteln/setzen

obj        Datei-Deskriptor  
          Neues Dateitrennzeichen

int1

(optional)

Resultat byte Aktuelles Dateitrennzeichen

Mit dieser Anweisung wird das Trennzeichen beim Einlesen von Dateizeilen variabler Länge (siehe FsiRead()) abgefragt (ein Argument) oder gesetzt (zwei Argumente). Ein Endezeichen mit dem Wert 0 (Vorgabewert) deaktiviert die Erkennung von Endezeichen. Beim Einlesen wird das Endezeichen mitgelesen, aber nicht im Alphawert gespeichert.

Beispiel:

```
// KommatHandle->FsiMark(44); // Liest die Zeichen bis zum KommatHandle->FsiRead(tAlpha);
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Datei-Deskriptor (obj) ungültig

## Kontakt

obj -> FsiMonitorAdd(alpha1[,  
int2[, alpha3[, alpha4]]])



Verzeichnis überwachen

obj      Deskriptor der Verzeichnisüberwachung

alpha1 Name des zu überwachenden  
Verzeichnisses

Option (optional)

FsiMonFlagsSubDirs

Überwachung

der  
untergeordneten

int2

Verzeichnisse

FsiMonFlagsNoDirChanges Änderung an

Verzeichnissen

ignorieren

alpha3 nicht zu überwachende Dateien (optional) alpha4

zu überwachende Dateien (optional) Siehe Verwandte

Befehle, EvtFsiMonitor

Mit diesem Befehl wird das Verzeichnis bestimmt, das überwacht werden soll. Die Anweisung kann mehrmals aufgerufen werden, um verschiedene Verzeichnisse zu überwachen.

Als (obj) wird der Deskriptor verwendet, der von dem Befehl FsiMonitorOpen() zurückgegeben wurde.

Das zu überwachende Verzeichnis wird als (alpha1) übergeben. Das \_Sys-Objekt stellt über seine Path-Eigenschaften einige Systempfade zur Verfügung.

Die Angabe der weiteren Parameter ist optional. Werden keine weiteren Parameter angegeben, wird bei jeder Änderung in dem angegebenen Verzeichnis das Ereignis EvtFsiMonitor aufgerufen.

Mit der Option FsiMonFlagsSubDirs können auch die untergeordnete Verzeichnisse mit überwacht werden.

Mit den Parametern (alpha3) und (alpha4) können bestimmte Dateien aus der Überwachung aus- bzw. eingeschlossen werden. Dabei können auch mehrere Dateimasken durch ein Semikolon getrennt angegeben werden. Die Überwachung wird mit der Anweisung FsiMonitorControl() gestartet bzw. angehalten.

Beispiele:

```
// Alle Änderungen in den "Eigenen Dateien" überwachenFsiMonitor->FsiMonitorAdd(_Sys->spPathMyDo
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der angegebene Deskriptor ist kein Deskriptor einer Überwachung oder ist ungültig.

obj -> FsiMonitorClose()  Überwachung externer

Verzeichnisse beenden

obj Deskriptor der

Verzeichnisüberwachung

Siehe Verwandte Befehle.

FsiMonitorOpen()

Mit dieser Anweisung wird eine mit FsiMonitorOpen() eingerichtete Verzeichnisüberwachung beendet. Der übergebene Deskriptor entspricht dem Rückgabewert des Befehls FsiMonitorOpen().

Der Diskriptor ist anschließend nicht mehr gültig und das Ereignis EvtFsiMonitor wird für das Fenster nicht mehr ausgeführt.

Mögliche Laufzeitfehler:

**ErrHdlInvalid** Der angegebene Deskriptor ist kein Deskriptor einer Überwachung oder ist ungültig.

## Kontakt

obj -> FsiMonitorControl(int1) : int 

Verzeichnisüberwachung starten / anhalten

obj Deskriptor der

Verzeichnisüberwachung

Optionen

FsiMonitorStart Überwachung

int1 starten

FsiMonitorStop Überwachung

beenden

Resultat int Fehlerwert

Siehe Verwandte Befehle,

EvtFsiMonitor, FsiMonitorAdd()

Mit dieser Anweisung kann die Verzeichnisüberwachung aktiviert oder deaktiviert werden. Als (obj) wird der von FsiMonitorOpen() zurückgegebene Deskriptor übergeben. Zur Steuerung stehen folgende Konstanten zur Verfügung:

FsiMonitorStart Startet die Verzeichnisüberwachung

FsiMonitorStop Setzt die Verzeichnisüberwachung aus.

Die Überwachung eines Verzeichnisses kann erst dann gestartet werden, wenn mindestens einmal die Anweisung FsiMonitorAdd() aufgerufen wurde.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der angegebene Deskriptor ist kein Deskriptor einer Überwachung oder ist ungültig.

ErrFsi... Fehler beim Zugriff auf ein überwachtes Verzeichnis

## Kontakt

obj -> FsiMonitorOpen(int1, int2) : handle  Überwachung  
externer Verzeichnisse einrichten

obj        Deskriptor des Fenster-Objekts

int1       Verzögerung vor dem ersten Ereignis-Aufruf

int2       Verzögerung vor jedem weiteren Aufruf

Resultat handle Deskriptor der Verzeichnisüberwachung 

Siehe Verwandte Befehle, EvtFsiMonitor,  
FsiMonitorClose(), FsiMonitorControl(),  
Verzeichnissüberwachung (Blog)

Mit diesem Befehl wird eine Verzeichnisüberwachung eingerichtet. Das zu überwachende Verzeichnis wird mit dem Befehl FsiMonitorAdd() angegeben. Bei einer Änderung in diesem Verzeichnis wird dann das Ereignis EvtFsiMonitor aufgerufen. Das Starten und Anhalten einer Verzeichnisüberwachung wird über die Funktion FsiMonitorControl() vorgenommen.

Im Parameter (obj) wird das Fenster angegeben, bei dem das EvtFsiMonitor-Ereignis aufgerufen werden soll. In den Parametern (int1) und (int2) werden zwei Verzögerungszeiten in Millisekunden angegeben. (int1) ist dabei die Verzögerung bis das Ereignis aufgerufen wird. Anschließend wird das Ereignis für jede Änderung in dem Verzeichnis aufgerufen. Darauf folgende Aufrufe werden erst nach der Verzögerung (int2) ausgeführt.


### Beispiel:

```
tFsiMonitor # tHdlFrame->FsiMonitorOpen(1000, 3000);
```

In diesem Fall wird eine Sekunde nach der Änderung im Verzeichnis das Ereignis aufgerufen. Werden 100 Dateien in das Verzeichnis kopiert, werden nach einer Sekunde für jede Datei die in dieser Sekunde kopiert werden konnte, das EvtFsiMonitor aufgerufen. Anschließend erfolgen die Aufrufe nur noch alle drei Sekunden.

### Mögliche Laufzeitfehler:

**ErrHdlInvalid** Der angegebene Deskriptor (obj) ist kein Fenster-Deskriptor oder ist ungültig.

obj -> FsiMonitorRemove(alpha1)  Verzeichnis aus  
der Überwachung entfernen

obj            Deskriptor der  
              Verzeichnisüberwachung  
alpha1 überwachtes Verzeichnis

Verwandte Befehle,

Siehe FsiMonitorAdd(),  
FsiMonitorControl()

Mit dieser Anweisung kann ein Verzeichnis wieder aus den überwachten Verzeichnissen ausgenommen werden.

Wurde ein Verzeichnis mit der Anweisung FsiMonitorAdd() zur Überwachung hinzugefügt, kann das Verzeichnis mit der Anweisung FsiMonitorRemove() wieder entfernt werden.



Nach einer Änderung muss die Überwachung neu gestartet werden.  
Beispiel:

```
tFsiMonitor # tHdlFrame->FsiMonitorOpen(1000, 3000);tFsiMonitor->FsiMonitorAdd(_Sys->spPathMyDocu
```

Mögliche Laufzeitfehler:

**ErrHdlInvalid** Der angegebene Deskriptor (obj) ist kein Deskriptor einer Überwachung oder ist ungültig.

**FsiOpen(alpha1,**   
**int2) : handle Datei**  
**öffnen**

**alpha1**    **Dateiname/-pfad**  
**int2**       **Optionen (siehe Text)**

**Resultat** **handle** **Datei-Deskriptor**    ☐  
   **oder Fehlerwert**

**Siehe**       **Verwandte Befehle,**  
              **FsiClose(), Fehlerwerte,**  
              **Beispiel**

Mit dieser Funktion wird die externe Datei (alpha1) geöffnet. In (alpha1) kann der vollständige Pfad angegeben werden. In (int2) muss mindestens eine der \_FsiAcs...-Optionen angegeben werden, da ansonsten keine Operationen auf der Datei durchgeführt werden können.

Außerdem sollte unbedingt eine der \_FsiDeny...-Optionen benutzt werden, um die Multiuser-Fähigkeit sicherzustellen.

Folgende Optionen können in (int2) angegeben werden:

- **FsiAcsR**

Nur Lesezugriff

- **FsiAcsW**

Nur Schreibzugriff

- **FsiAcsRW**

Lese- und Schreibzugriff

- **FsiDenyR**

Exklusiver Lesezugriff

- **FsiDenyW**

Exklusiver Schreibzugriff

- **FsiDenyRW**

Exklusiver Lese- und Schreibzugriff

- **FsiDenyNone**

Kein exklusiver Zugriff

- **FsiCreate**

Datei anlegen oder öffnen

- **FsiCreateNew Datei**

explizit anlegen

- **FsiTruncate**

Datei leeren

## Kontakt

- FsiAppend Datei  
erweitern
- FsiSyncWrite  
Synchrones Schreiben
- FsiStdRead Standard-  
Lesemodus
- FsiStdWrite Standard-  
Schreibmodus
- FsiBuffer Lesen/Schreiben  
puffern
- FsiNoCache Lesen/Schreiben nicht  
puffern
- FsiANSI  
Lesen/Schreiben einer ANSI-Datei
- FsiPure  
Keine Zeichenwandlung beim Lesen oder Schreiben einer Datei
- FsiNameC16  
Dateiname ist im CONZEPT 16-Zeichensatz angegeben
- FsiNameUtf8  
Dateiname ist im UTF-8-Zeichensatz angegeben

Die folgenden Optionen (int2) gelten nur für Windows:

- FsiDeleteOnClose Datei  
temporär öffnen

Die folgenden Optionen (int2) gelten nur für UNIX:

- FsiUserR  
Leseberechtigung für Benutzer
- FsiUserW  
Schreibberechtigung für Benutzer
- FsiGroupR Leseberechtigung für  
Gruppe
- FsiGroupW Schreibberechtigung für  
Gruppe



## Kontakt

- FsiOtherR Leseberechtigung für  
Andere
- FsiOtherW Schreibberechtigung für  
Andere

Das Resultat ist größer Null, wenn die Datei erfolgreich geöffnet wurde. Bei einem negativen Resultat ist ein Fehler aufgetreten. Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden. Wurde die Option FsiCreateNew angegeben und die Datei existiert bereits, ist das Resultat ErrFsiExists.

### Beispiele:

```
tHandle # FsiOpen('\Test.asc', _FsiStdRead);if (tHandle > 0){ ... tHandle->FsiClose();}else{ /  
  
// Liste aller Dateien eines Verzeichnisses erstellensub WriteFileList( aPath : alpha(4096);)
```

**FsiPath([int1]) : alpha** Aktuelles



**Verzeichnis ermitteln**

**Optionen (optional)**

**FsiNameC16** Aktuelles Verzeichnis wird

im

**CONZEPT 16-Zeichensatz**

**int1**

ermittelt (Standard)

**FsiNameUtf8** Aktuelles Verzeichnis wird

im UTF-8-Zeichensatz

ermittelt

**Resultat alpha** Aktuelles Verzeichnis

**Verwandte Befehle, FsiPathChange(),**

Siehe

**FsiSplitName()**

Diese Funktion liefert das aktuelle Verzeichnis inklusive Laufwerksbuchstabe zurück.

Wird im optionalen Argument (int1) **FsiNameUtf8** angegeben, wird das aktuelle Verzeichnis als UTF-8-Zeichenkette ermittelt.

**Mögliche Laufzeitfehler:**

**ErrValueInvalid** Es wurde eine ungültige Option (int1) angegeben.

## Kontakt

**FsiPathChange(alpha1[,  
int2]) : int**



**Verzeichnis wechseln**

**alpha1** Neues Verzeichnis

**Optionen (optional)**

**\_FsiNameC16** Neues Verzeichnis

(alpha1) wird im  
CONZEPT 16-Zeichensatz

**int2** erwartet (Standard)

**\_FsiNameUtf8** Neues Verzeichnis

(alpha1) wird im  
UTF-8-Zeichensatz

erwartet

**Resultat int** Fehlerwert



**Verwandte Befehle**, **FsiPath()**,

Siehe

**Fehlerwerte**

Diese Funktion ändert das aktuelle Verzeichnis auf (alpha1).

Wird im optionalen Argument (int2) **\_FsiNameUtf8** angegeben, wird das neue Verzeichnis (alpha1) als UTF-8-Zeichenkette erwartet. Somit ist es auch möglich in Verzeichnisse mit Umlauten anderer Sprachen zu wechseln.

Die Anweisung gibt einen Fehlerwert (**\_ErrFsi...**) zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft **FsiError** abgefragt werden.

```
// Wechseln erfolgreich if (FsiPathChange('C:\') = _ErrOk){ ...}
```

**Mögliche Laufzeitfehler:**

**\_ErrValueInvalid** Es wurde eine ungültige Option (int2) angegeben.

## Kontakt

**FsiPathCreate(alpha1[,  
int2]) : int**



**Verzeichnis erstellen**

**alpha1** Verzeichnisname

**Optionen (optional)**

**FsiNameC16** Verzeichnisname (alpha1)

wird im

**CONZEPT 16-Zeichensatz**

**int2** erwartet (Standard)

**FsiNameUtf8** Verzeichnisname (alpha1)

wird im

**UTF-8-Zeichensatz**

erwartet

**Resultat int** Fehlerwert



**Verwandte Befehle**, **FsiPathChange()**,

Siehe

**FsiPathDelete()**, **Fehlerwerte**

Diese Funktion legt ein neues Verzeichnis mit dem Namen (alpha1) an. Die Anweisung gibt einen Fehlerwert (**\_ErrFsi...**) zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft **FsiError** abgefragt werden.

Wird im optionalen Argument (int2) **FsiNameUtf8** angegeben, wird der

Verzeichnisname (alpha1) als UTF-8-Zeichenkette erwartet. Somit ist es auch möglich Verzeichnisse mit Umlauten anderer Sprachen zu erstellen.



Es kann immer nur eine Ebene gleichzeitig angelegt werden.  
**Mögliche Laufzeitfehler:**

**\_ErrValueInvalid** Es wurde eine ungültige Option (int2) angegeben.

## Kontakt

**FsiPathDelete(alpha1[,  
int2]) : int**



**Verzeichnis löschen**

**alpha1** Verzeichnisname

**Optionen (optional)**

**FsiNameC16** Verzeichnisname (alpha1)

wird im

**CONZEPT 16-Zeichensatz**

**int2**

erwartet (Standard)

**FsiNameUtf8** Verzeichnisname (alpha1)

wird im

**UTF-8-Zeichensatz**

erwartet

**Resultat int** Fehlerwert



**Verwandte Befehle**, **FsiPathCreate()**,

Siehe

**FsiDelete()**, **Fehlerwerte**

Diese Funktion löscht das Verzeichnis mit dem Namen (alpha1). Dateien können mit **FsiDelete()** gelöscht werden.

Wird im optionalen Argument (int2) **FsiNameUtf8** angegeben, wird der Verzeichnisname (alpha1) als UTF-8-Zeichenkette erwartet. Somit ist es auch möglich Verzeichnisse mit Umlauten anderer Sprachen zu löschen.



Es können nur leere Verzeichnisse gelöscht werden.

Die Anweisung gibt einen Fehlerwert (**ErrFsi...**) zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft **FsiError** abgefragt werden.

**Mögliche Laufzeitfehler:**

**ErrValueInvalid** Es wurde eine ungültige Option (int2) angegeben.

obj ->

**FsiRead**(var1[,  
int2]) : int



Dateibereich lesen

obj      Datei-Deskriptor  
var1      Speicherziel (Feld,  
            Variable oder Array)  
int2      Bereichsgröße  
            (optional)

**Resultat** int Bereichsgröße oder Fehlerwert 

Verwandte Befehle,

Siehe FsiOpen(), FsiSeek(),  
FsiWrite(), Fehlerwerte

Mit dieser Funktion werden Daten aus der externen Datei (obj) ab der aktuellen Position gelesen (siehe FsiSeek()). In (var1) muss ein Datenbankfeld, eine Variable oder ein Array angegeben werden. Arrays aus Alphafeldern sind dabei nicht zulässig. Sofern (int2) nicht angegeben ist, wird die der Größe der Variablen (var1) entsprechende Anzahl von Bytes eingelesen. Der Wert in (int2) kann daher auch nicht größer als die Variable selbst sein.

Das Resultat gibt die Anzahl der gelesenen Bytes zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert (\_ErrFsi...). Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.

Je nach Datentyp werden folgende Formate benutzt:

- alpha (Alphanumerisch)

Es kann sowohl eine feste als auch eine variable Anzahl von Zeichen eingelesen werden. Bei einer festen Anzahl steht in (int2) die Anzahl der zu lesenden Bytes. Bei einer variablen Anzahl wird (int2) weggelassen. In diesem Fall werden soviele Zeichen eingelesen, bis das Endezeichen (siehe FsiMark()), die maximale Variablenlänge oder das Dateiende erreicht ist. Das Endezeichen selbst wird zwar gelesen, aber nicht in die Variable übertragen. Der Dateizeiger steht dann auf dem ersten Byte hinter dem Endezeichen. Das ASCII- Zeichen NUL kann nicht eingelesen werden.

Beim Lesen werden die Zeichenketten standardmäßig von der OEM-Zeichencodierung in die interne Zeichencodierung gewandelt. Durch entsprechende Angaben bei der Anweisung FsiOpen() kann eine andere Wandlung vorgenommen oder die Wandlung verhindert werden.

- float (Gleitkomma)

Es werden acht Bytes gelesen. Das Format entspricht dem IEEE-Double-Precision-Real-Format (64 Bit).

- word (Ganzzahlig kurz)

## Kontakt

Es werden zwei Bytes gelesen. Das Format entspricht dem Intel-Wortformat (16 Bit - little endian).

- int (Ganzzahlig lang)

Es werden vier Bytes gelesen. Das Format entspricht dem Intel-Doppelwortformat (32 Bit - little endian).

- date (Datum)

Es werden vier Bytes gelesen. (Byte 1 = Tag, Byte 2 = Monat, Byte 3 = Jahr, Byte 4 ist grundsätzlich leer.)

- time (Zeit)

Es werden vier Bytes gelesen. (Byte 1 = Stunde, Byte 2 = Minute, Byte 3 = Sekunde, Byte 4 = Hundertstelsekunde.)

- logic (Logisch)

Es wird ein Byte gelesen (Wert gleich 0 entspricht false, Wert gleich 1 entspricht true.)



Beim Einlesen von ASCII-Dateien wird ausschließlich mit alphanumerischen Feldern bzw. Variablen gearbeitet. Diese müssen dann gegebenenfalls in der Prozedur in andere Feldtypen umgewandelt werden.

Beispiele:

```
// Einlesen eines ASCII-Werts fester Länge: HdL->FsiRead(tAlpha, 20); // Einlesen eines ASCII-Werts
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ist ungültig.

## Kontakt

obj -> FsiReadMem(handle1, int2,   
int3) : int

Datei in Memory-Objekt lesen


obj        Datei-Deskriptor

handle1 Deskriptor des Memory-Objekts

int2       Position im Memory-Objekt

int3       Anzahl der Bytes

Anzahl der gelesenen Bytes

**Resultat int**  oder Fehlerwert

Siehe       Verwandte Befehle, FsiWriteMem()

Mit dieser Funktion werden Daten aus der externen Datei (obj) ab der aktuellen Position gelesen (siehe FsiSeek() bzw. FsiSeek64()). In (handle1) muss der Deskriptor eines Memory-Objekts angegeben werden. Aus der Datei werden maximal (int3) Bytes gelesen und ab der Position (int2) in das Memory-Objekt übertragen. Gegebenenfalls wird der Wert der Eigenschaft Len erhöht.

Werden aus der externen Datei Zeichenketten in das Objekt gelesen, muss die Eigenschaft Charset des Memory-Objekts auf den Zeichensatz der externen Datei gesetzt werden, damit die Zeichenketten korrekt verarbeitet werden können. Eine Konvertierung der Zeichenkodierung aufgrund der Angaben bei FsiOpen() findet nicht statt.

Das Resultat gibt die Anzahl der gelesenen Bytes zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert (\_ErrFsi...). Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.

### Mögliche Laufzeitfehler:

\_ErrHdlInvalid       Der Datei-Deskriptor (obj) oder der Deskriptor des Memory-Objekts ist ungültig.

\_ErrValueRange Die übergebenen Werte in (int2) oder (int3) sind ungültig.



**FsiRename(alpha1,**



**alpha2) : int**

**Datei umbenennen**

Alter

**alpha1**

Datei-/pfadname

**alpha2**

Neuer

Datei-/pfadname

**Resultat int Fehlerwert**  —

**Verwandte**

**Siehe**

**Befehle,**

**Fehlerwerte**

Durch Umbenennung ist auch ein Verschieben einer Datei in ein anderes Verzeichnis (nur auf demselben Laufwerk) möglich.

Die Anweisung gibt einen Fehlerwert (**\_ErrFsi...**) zurück. Der Fehlerwert des Betriebssystems kann über die Eigenschaft **FsiError** abgefragt werden.

**Beispiel:**

```
if (FsiRename('C:\Src\File.dat', 'C:\Tar\MovedFile.dat') = _ErrOk){ ...}
```

obj -> FsiSeek([int1]) : int 

Dateizeigerposition ermitteln/setzen

obj      Datei-Deskriptor  
 int1     Neue Dateizeiger-Position

(optional)

Resultat int Aktuelle Dateizeigerposition

Verwandte Befehle,

Siehe    FsiSeek64(), FsiRead(),  
FsiWrite()

Diese Funktion dient zur Abfrage (ein Argument) oder zum Setzen (zwei Argumente) des Dateizeigers in der externen Datei (obj). Ab dieser Position wird anschließend gelesen (siehe FsiRead()) oder geschrieben (siehe FsiWrite()). Die Positionsangabe erfolgt in Bytes - das erste Byte einer Datei steht an Position 0. FsiSeek() kann nicht bei Dateien benutzt werden, die mit der Option FsiBuffer bei FsiOpen() geöffnet wurden.

Im Resultat wird die angegebene Position zurückgegeben.



Ab einer Dateigröße von 2 GB muss der Befehl FsiSeek64() verwendet werden. Mit dem Befehl FsiSeek() kann der Dateizeiger auch hinter das Ende einer Datei positioniert werden. Bei einem anschließenden FsiWrite() wird die Datei auf die entsprechende Größe vergrößert. Dies kann ebenfalls mit dem Befehl FsiSize() erfolgen.

Beispiel:

```
// Dateizeiger an Dateianfang setzentHandle->FsiSeek(0);
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ungültig

obj -> FsiSeek64([bigint1]) : bigint 

Dateizeigerposition ermitteln/setzen (64 Bit)

obj      Datei-Deskriptor  
          Neue Dateizeiger-Position  
bigint1

(optional)

Resultat bigint Aktuelle Dateizeigerposition  
          Verwandte Befehle, FsiSeek(),

Siehe

FsiRead(), FsiWrite()

Diese Funktion dient zur Abfrage (ein Argument) oder zum Setzen (zwei Argumente) des Dateizeigers in der externen Datei (obj). Ab dieser Position wird anschließend gelesen (siehe FsiRead()) oder geschrieben (siehe FsiWrite()). Die Positionsangabe erfolgt in Bytes - das erste Byte einer Datei steht an Position 0. FsiSeek64() kann nicht bei Dateien benutzt werden, die mit der Option FsiBuffer bei FsiOpen() geöffnet wurden.

Im Resultat wird die angegebene Position zurückgegeben.



Im Gegensatz zu dem Befehl FsiSeek() kann dieser Befehl auch bei Dateien mit mehr als 2 GB Größe verwendet werden.

Mit dem Befehl FsiSeek64() kann der Dateizeiger auch hinter das Ende einer Datei positioniert werden. Bei einem anschließenden FsiWrite() wird die Datei auf die entsprechende Größe vergrößert. Dies kann ebenfalls mit dem Befehl FsiSize64() erfolgen.

Beispiel:

```
// Dateizeiger an Dateianfang setzentHandle->FsiSeek64(0);
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ist ungültig.

obj -> FsiSize([int1]) : int 

Dateigröße ermitteln/setzen

obj      Datei-Deskriptor  
           Neue Dateigröße

int1

(optional)

Resultat int Aktuelle Dateigröße  
Verwandte Befehle,

Siehe

FsiSize64(), FsiOpen()

Mit dieser Funktion wird die Größe der externen Datei (obj) abgefragt (ein Argument) oder geändert (zwei Argumente). Sofern die Datei vergrößert wird, ist der Inhalt zwischen dem alten und dem neuen Dateiende undefiniert. Beim Ändern der Dateigröße muss die Datei mit Schreibberechtigung geöffnet sein. Das Ändern der Größe ist nicht bei Dateien möglich, die mit der Option FsiBuffer bei FsiOpen() geöffnet wurden. In diesem Fall liefert die Abfrage die Dateigröße zum Zeitpunkt von FsiOpen() zurück.

Das Resultat gibt die aktuelle Dateigröße in Byte zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.



Ab einer Dateigröße von 2 GB muss der Befehl FsiSize64() verwendet werden. FsiSize() kann auch zur Abfrage der Dateigröße eines Verzeichniseintrags benutzt werden (siehe FsiDirRead()). Dazu wird in (obj) der Deskriptor des Verzeichnisses übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ist ungültig.

obj -> FsiSize64([bigint1]) : bigint 

Dateigröße ermitteln/setzen (64 Bit)

obj      Datei-Deskriptor  
          Neue Dateigröße

**bigint1**  
(optional)

Resultat bigint Aktuelle Dateigröße  
          Verwandte Befehle,

Siehe

FsiSize(), FsiOpen()

Mit dieser Funktion wird die Größe der externen Datei (obj) abgefragt (ein Argument) oder geändert (zwei Argumente). Sofern die Datei vergrößert wird, ist der Inhalt zwischen dem alten und dem neuen Dateiende undefiniert. Beim Ändern der Dateigröße muss die Datei mit Schreibberechtigung geöffnet sein. Das Ändern der Größe ist nicht bei Dateien möglich, die mit der Option FsiBuffer bei FsiOpen() geöffnet wurden. In diesem Fall liefert die Abfrage die Dateigröße zum Zeitpunkt von FsiOpen() zurück.

Das Resultat gibt die aktuelle Dateigröße in Byte zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.



Im Gegensatz zu dem Befehl FsiSize() kann dieser Befehl auch bei Dateien mit mehr als 2 GB Größe verwendet werden.

FsiSize64() kann auch zur Abfrage der Dateigröße eines Verzeichniseintrags benutzt werden (siehe FsiDirRead()). Dazu wird in (obj) der Deskriptor des Verzeichnisses übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ist ungültig.

**FsiSplitName(alpha1,  
int2) : alpha**



**Dateinamen zerlegen**

**alpha1** Dateiname/-pfad

**Optionen**

**FsiNamePNE** Laufwerk,

Pfad, Datei  
und  
Erweiterung  
ermitteln

**FsiNamePN** Laufwerk,

Pfad und  
Datei  
ermitteln

**FsiNameP** Laufwerk

und Pfad mit  
Separator

**int2**

ermitteln

**FsiNameNE** Datei und  
Erweiterung  
ermitteln

**FsiNameN** Datei  
ermitteln

**FsiNameE** Erweiterung  
ermitteln

**FsiNamePP** Laufwerk  
und Pfad  
ohne  
Separator  
ermitteln

Teil des

**Resultat alpha** Dateinamens/-pfades

Verwandte Befehle,

Siehe

**FsiDirRead()**, **FsiPath()**

Mit dieser Funktion kann ein qualifizierter Dateiname (alpha1) in seine Bestandteile zerlegt werden. Je nach Option in (int2) wird der entsprechende Namensteil extrahiert und als Resultat zurückgeliefert.

## Beispiele

```
FsiSplitName('C:\DIR\FILE.EXT', _FsiNamePNE); // 'C:\DIR\FILE.EXT'FsiSplitName('C:\DIR\FILE.EXT',
```



Es können bis zu 4096 Zeichen verarbeitet werden. Längere Angaben werden bei 4096 Zeichen abgeschnitten.

## Kontakt

obj -> FsiStamp(int1[, bigint2]) : bigint  Dateiinformationen  
ermitteln/setzen (Zeitstempel)

obj      Datei-Deskriptor  
         Informationstyp  
         FsiDtModified Letzten  
                 Änderungszeitstempel

                                 ermitteln/setzen  
int1      FsiDtAccessed Letzten  
                 Zugriffszeitstempel

                                 ermitteln/setzen  
                 FsiDtCreated Erstellungszeitstempel  
                                 ermitteln/setzen

bigint2   Neuer Zeitstempel (optional)

Resultat bigint Aktueller Zeitstempel

Siehe Verwandte Befehle, FsiOpen(),

FsiDate(), FsiTime()

Mit dieser Funktion können Zeitstempel in UTC-Zeit der externen Datei (obj) abgefragt (zwei Argumente) oder geändert werden (drei Argumente).

Bei FAT-Dateisystemen ist nur der Zeitstempel der letzten Änderung verfügbar. Bei Linux-Dateisystemen ist der Zeitstempel des letzten Zugriffs nicht verfügbar. Bei NTFS-Dateisystemen sind alle Zeitstempel verfügbar. Sofern ein Zeitstempel nicht verfügbar ist, wird 0\b zurückgeliefert.

FsiStamp() kann auch zur Abfrage der Zeitstempel eines Verzeichniseintrags benutzt werden (siehe FsiDirRead()). Dazu wird in (obj) der Deskriptor des Verzeichnisses übergeben.

Der Zeitstempel kann mit CnvCB() in einen calttime-Wert konvertiert werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Datei-Deskriptor (obj) ungültig

obj -> FsiTime(int1[, time2]) : time  Dateiinformationen

ermitteln/setzen (Zeitwerte)

obj      Datei-Deskriptor

Informationstyp

FsiDtModified Letzte

Änderungsuhrzeit

ermitteln/setzen

int1      FsiDtAccessed Letzte

Zugriffsuhrzeit

ermitteln/setzen

FsiDtCreated Erstellungsuhrzeit

ermitteln/setzen

time2      Neue Uhrzeit (optional)

Resultat time      Aktuelle Uhrzeit

Verwandte Befehle, FsiOpen(),

Siehe FsiDate(), FsiStamp()

Mit dieser Funktion können Zeitwerte der externen Datei (obj) ermittelt (zwei Argumente) oder gesetzt werden (drei Argumente).

Bei FAT-Dateisystemen ist nur die Uhrzeit der letzten Änderung verfügbar. Bei Linux-Dateisystemen ist die Uhrzeit des letzten Zugriffs nicht verfügbar. Bei NTFS-Dateisystemen sind alle Zeitwerte verfügbar. Sofern ein Zeitwert nicht verfügbar ist, wird 0:0:0.0 zurückgeliefert.

FsiTime() kann auch zur Abfrage der Zeitwerte eines Verzeichniseintrags benutzt werden (siehe FsiDirRead()). Dazu wird in (obj) der Deskriptor des Verzeichnisses übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ist ungültig.



## Kontakt

obj -> FsiWrite(var1[,  
int2]) : int



Dateibereich schreiben

obj        Datei-Deskriptor

var1       Speicherquelle

int2       Bereichsgröße (optional)  
            Anzahl geschriebener

Resultat int Zeichen oder  
            Fehlerwert



Verwandte Befehle,  
FsiOpen(), FsiRead(),

Siehe

FsiSeek(), Fehlerwerte,  
Beispiel

Mit dieser Funktion werden Daten in die externe Datei (obj) ab der aktuellen Position geschrieben (siehe FsiSeek()). In (var1) kann ein Datenbankfeld, eine Variable, ein Array oder eine Zeichenketten-Konstante angegeben werden. Arrays aus Alphafeldern sind dabei nicht zulässig. Sofern (int2) nicht angegeben ist, wird die der Größe der Variablen (var1) entsprechende Anzahl von Bytes geschrieben. Der Wert in (int2) kann daher auch nicht größer als die Variable selbst sein.

Leerzeichen am Ende eines Alphawerts bleiben erhalten. Für die Erstellung von ASCII-Dateien muss ausschließlich mit alphanumerischen Werten gearbeitet werden. Andere Feldtypen müssen in der Prozedur entsprechend umgewandelt werden.

Beim Schreiben werden die Zeichenketten standardmäßig von der internen Zeichencodierung in die OEM-Zeichencodierung gewandelt. Durch entsprechende Angaben bei der Anweisung FsiOpen() kann eine andere Wandlung vorgenommen oder die Wandlung verhindert werden.

Sollen in eine externe Datei Nullen geschrieben werden, kann dafür keine Zeichenkette verwendet werden. Statt dessen kann zum Beispiel ein Array vom Typ byte verwendet werden. Da immer das gesamte Array geschrieben wird, werden auch die auf Null gesetzten Zellen geschrieben.

Das Resultat gibt die Anzahl der geschriebenen Bytes zurück. Ist das Resultat negativ ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert (\_ErrFsi...). Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.

Beispiel:

```
// Kopieren einer Dateidefine{ sMem                    : _Mem8K // 8 Kilobytes}sub FsiCopy( aSrc tDstHdl-  
// Datum & Uhrzeit wiederherstellen                >FsiDate(_FsiDtModified, FsiDate(tSrcHdl, _Fsi
```

Mögliche Laufzeitfehler:

\_ErrHdlInvalid Datei-Deskriptor (obj) ungültig

## Kontakt

obj -> FsiWriteMem(handle1, int2,  
int3) : int



Memory-Objekt in Datei schreiben

obj        Datei-Deskriptor

handle1    Deskriptor des Memory-Objekts

int2        Position im Memory-Objekt

int3        Anzahl der zu schreibenden Bytes

**Resultat** int    Anzahl der geschriebenen Bytes



oder Fehlerwert

Siehe       Verwandte Befehle, FsiOpen()

Mit dieser Funktion werden Daten in die externe Datei (obj) ab der aktuellen Position geschrieben (siehe FsiSeek() bzw. FsiSeek64()). In (handle1) muss der Deskriptor eines Memory-Objekts angegeben werden. Aus dem Memory-Objekt werden ab der Position (int2) eine Anzahl von (int3) Bytes in die Datei geschrieben.

Sollen Zeichenketten in die externen Datei geschrieben, muss die Eigenschaft Charset des Memory-Objekts auf den Zeichensatz der externen Datei gesetzt werden, damit die Zeichenketten beim Einfügen in das Memory-Objekt (siehe MemWriteStr()) korrekt verarbeitet werden können. Eine Konvertierung der Zeichenkodierung aufgrund der Angaben bei FsiOpen() findet nicht statt.

Das Resultat gibt die Anzahl der geschrieben Bytes zurück. Ist das Resultat negativ ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert ( ErrFsi...). Der Fehlerwert des Betriebssystems kann über die Eigenschaft FsiError abgefragt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid

Datei-Deskriptor (obj) oder der Deskriptor des Memory-Objekts ist ungültig.

ErrValueRange

Der in (int2) oder (int3) übergebene Wert ist außerhalb des zulässigen Bereichs.

Konstanten für Dateibefehle (Extern)

Konstanten für Dateibefehle (Extern)

Dateibefehle

Siehe (Extern)

- ComprFmtDeflate
- ComprFmtGzip
- ComprFmtZlib
- ComprLvlDefault
- FsiAcsR
- FsiAcsRW
- FsiAcsW
- FsiANSI
- FsiAppend
- FsiAttrArchive
- FsiAttrDir
- FsiAttrExec
- FsiAttrHidden
- FsiAttrRead
- FsiAttrSystem
- FsiAttrWrite
- FsiBuffer
- FsiCompressFast
- FsiCompressMed
- FsiCompressSlow
- FsiCompressStd
- FsiCreate
- FsiCreateNew
- FsiDecode
- FsiDeleteOnClose
- FsiDenyNone
- FsiDenyR
- FsiDenyRW
- FsiDenyW
- FsiDiskAvailMB
- FsiDiskExists
- FsiDiskFree
- FsiDiskFreeMB
- FsiDiskReady
- FsiDiskTotal
- FsiDiskTotalMB
- FsiDtAccessed
- FsiDtCreated
- FsiDtModified
- FsiEncrypt
- FsiFileCRC32
- FsiFileMD5
- FsiFileRMD160
- FsiFileSHA1
- FsiFileSHA256
- FsiFileSHA384

- FsiFileSHA512
- FsiFileVersion
- FsiFileVersionHex
- FsiGroupR
- FsiGroupW
- FsiMonActionCreate
- FsiMonActionDelete
- FsiMonActionModify
- FsiMonActionRename
- FsiMonFlagsSubDirs
- FsiMonitorStart
- FsiMonitorStop
- FsiNameC16
- FsiNameE
- FsiNameN
- FsiNameNE
- FsiNameP
- FsiNamePN
- FsiNamePNE
- FsiNamePP
- FsiNameUtf8
- FsiNoCache
- FsiOtherR
- FsiOtherW
- FsiPure
- FsiStdRead
- FsiStdWrite
- FsiSyncWrite
- FsiTruncate
- FsiUserR
- FsiUserW

## Kontakt

**\_FsiAcsR**

**Nur Lesezugriff**

**Wert** <sup>1/</sup>

**0x00000001**

**Verwandte**

**Siehe Befehle,**

**FsiOpen()**

**Option bei FsiOpen() durch die eine Datei mit ausschließlichem Lesezugriff geöffnet werden kann.**

**Auf eine mit ausschließlichem Lesezugriff geöffnete Datei sind keine Schreiboperationen möglich.**

**\_FsiAcsRW**

**Lese- und Schreibzugriff**

**Wert** <sup>3/</sup>

**0x00000003**

**Verwandte**

**Siehe Befehle,**

**FsiOpen()**

**Option bei FsiOpen() durch die eine Datei mit Lese- und Schreibzugriff geöffnet werden kann.**

## Kontakt

**\_FsiAcsW**

**Nur Schreibzugriff**

**Wert**

**0x00000002**

**Verwandte**

**Siehe Befehle,**

**FsiOpen()**

**Option bei FsiOpen() durch die eine Datei mit ausschließlichem Schreibzugriff geöffnet werden kann.**

**Auf eine mit ausschließlichem Schreibzugriff geöffnete Datei sind keine Leseoperationen möglich.**

**\_FsiANSI**

**Lesen/Schreiben einer ANSI-Datei**

Wert 33.554.432 /  
**0x02000000**

**Verwandte**

**Befehle,**

Siehe **FsiOpen()**,

**FsiRead()**,

**FsiWrite()**

Option bei **FsiOpen()** durch die eine automatische Zeichenwandlung beim Lesen (siehe **FsiRead()**) bzw. Schreiben (siehe **FsiWrite()**) durchgeführt werden kann. Die Option kann nicht mit **\_FsiPure** kombiniert werden.



**\_FsiAppend**

**Datei erweitern**

Wert <sup>64</sup> /

**0x00000040**

**Verwandte**

Siehe Befehle,

**FsiOpen()**,

**FsiTruncate**

Option bei **FsiOpen()** durch die eine Datei erweitert werden kann. Der Dateizeiger wird auf das Dateiende positioniert. Die Datei muss bereits existieren, ansonsten wird diese Option ignoriert.

Diese Option ersetzt somit folgende Anweisung:

**FsiSeek(tHandle, FsiSize(tHandle))**

**\_FsiAttrArchive**

Archivdatei ermitteln/setzen

Wert 32 / 0x0020

**Verwandte**

Siehe **Befehle**,

**FsiAttributes()**

Option bei **FsiAttributes()**. Mit dieser Konstanten kann das Archiv-Attribut einer Datei ermittelt oder gesetzt werden.

**\_FsiAttrDir**

**Verzeichnis ermitteln**

**Wert 16 / 0x0010**

**Verwandte**

**Siehe Befehle,**

**FsiAttributes()**

**Option bei FsiAttributes(). Ist dieses Dateiaattribut gesetzt, handelt es sich um ein Verzeichnis.**

**Das Attribut kann nur gelesen werden.**

## Kontakt

\_FsiAttrExec

Ausführbarkeit ermitteln (nur UNIX)

Wert 1.024 / 0x0400

Verwandte

Siehe Befehle,

FsiAttributes()

Option bei FsiAttributes(). Ist dieses Attribut gesetzt, handelt es sich um eine ausführbare Datei. Das Dateiattribut "Executable" existiert nur unter Linux.

**\_FsiAttrHidden**

Versteckte Datei ermitteln/setzen

Wert 2 / 0x0002

**Verwandte**

Siehe Befehle,

FsiAttributes()

Option bei FsiAttributes() durch die das Datei-Attribut "Versteckt" ermittelt/gesetzt werden kann.

**\_FsiAttrRead**

**Leseberechtigung ermitteln**

**Wert 256 / 0x0100**

**Verwandte**

**Siehe Befehle,**

**FsiAttributes()**

**Option bei FsiAttributes(). Ist dieses Dateiattribut gesetzt, kann die Datei gelesen werden. Dies ist unter Windows-Betriebssystemen immer der Fall.**



**Mit dieser Anweisung werden nur die Datei-Attribute überprüft. Es werden keine Verzeichnis- oder Dateirechte berücksichtigt.**

**\_FsiAttrSystem**

Systemdatei ermitteln/setzen

Wert 4 / 0x0004

**Verwandte**

Siehe **Befehle**,

**FsiAttributes()**

Option bei **FsiAttributes()** durch die eine Markierung für eine Systemdatei ermittelt/gesetzt werden kann.

**\_FsiAttrWrite**

Schreibberechtigung ermitteln/setzen

Wert 512 / 0x0200

Verwandte

Siehe Befehle,

FsiAttributes()

Option bei FsiAttributes(). Ist dieses Dateiaattribut gesetzt, kann die Datei geschrieben werden.



Mit dieser Anweisung werden nur die Datei-Attribute überprüft. Es werden keine Verzeichnis- oder Dateirechte berücksichtigt.



### \_FsiBuffer

#### Gepuffertes Lesen/Schreiben

Wert 16.777.216 /  
0x01000000

#### Verwandte

Siehe Befehle,

#### FsiOpen()

Option bei FsiOpen() durch die die Lese- bzw. Schreibzugriffe auf eine Datei durch einen 8 KB großen Puffer beschleunigt werden können. Dies bringt Geschwindigkeitsvorteile beim Verarbeiten vieler kleiner, sequenzieller Lese- bzw. Schreiboperationen.

Dabei ist folgendes zu beachten:

- \_FsiBuffer ist nur wirksam in Verbindung mit \_FsiAcsR oder \_FsiAcsW. Dementsprechend ist die Pufferung dann für Schreiben oder Lesen aktiv.
- Beim gepufferten Betrieb können folgende Funktionen nicht oder nur eingeschränkt verwendet werden:

FsiSeek() ist nicht möglich.

FsiSize() Das Setzen einer neuen Dateigröße ist nicht möglich, die Abfrage der aktuellen Größe liefert die Größe zum Zeitpunkt von FsiOpen().

**\_ComprFmtDeflate**

**Komprimierung im DEFLATE-Format**

**Wert 0**

**FsiFileCompress(),**

Siehe **MemCompress(),**

**ComprFmtGzip,**

**ComprFmtZlib**

**Diese Konstante kann bei der Anweisung FsiFileCompress() und MemCompress() angegeben werden, um den Inhalt der Datei im DEFLATE-Format (siehe RFC 1951) zu komprimieren.**

**\_ComprFmtGzip**

**Komprimierung im GZIP-Format**

**Wert 1**

**FsiFileCompress()**,

Siehe **MemCompress()**,

**ComprFmtDeflate**,

**ComprFmtZlib**

Diese Konstante kann bei der Anweisung **FsiFileCompress()** und **MemCompress()** angegeben werden, um den Inhalt der Datei im GZIP-Format (siehe **RFC 1952**) zu komprimieren.

**\_ComprFmtZlib**

**Komprimierung im ZLIB-Format**

**Wert 2**

**FsiFileCompress(),**

Siehe **MemCompress(),**

**ComprFmtDeflate,**

**ComprFmtGzip**

Diese Konstante kann bei der Anweisung **FsiFileCompress()** und **MemCompress()** angegeben werden, um den Inhalt der Datei im ZLIB-Format (siehe **RFC 1950**) zu komprimieren.

## Kontakt

**\_ComprLvlDefault**

**Standard-Kompressionsstufe**

**Wert -1**

**Siehe FsiFileCompress(),**

**Mit dieser Konstante kann bei der Anweisung FsiFileCompress() und MemCompress() angegeben werden, dass die Standard-Kompressionsstufe verwendet werden soll.**

**\_FsiCreate**

**Datei anlegen oder öffnen**

Wert <sup>16</sup> /

**0x00000010**

**Verwandte**

**Befehle,**

Siehe **FsiOpen()**,

**FsiTruncate,**

**FsiAppend**

Option bei **FsiOpen()** durch die eine Datei mit der Größe 0 Byte angelegt werden kann.

Existiert die Datei bereits, wird sie erweiternd geöffnet (siehe **FsiAppend**).

Soll eine bestehende Datei überschrieben werden, muss die Option **FsiTruncate** verwendet werden.

**\_FsiCreateNew**

**Neue Datei explizit anlegen**

**Wert 268.435.456**

**/ 0x10000000**

**Verwandte**

**Siehe Befehle,**

**FsiOpen()**

**Option bei FsiOpen() durch die eine Datei explizit angelegt werden kann. Existiert die Datei bereits, wird der Fehlerwert ErrFsiExists zurückgegeben.**

**\_FsiDecode**

**Datei dekomprimieren und entschlüsseln**

**Wert 512 / 0x0200**

**Verwandte**

**Befehle,**

**Siehe FsiFileProcess(),**

**FsiFileMD5,**

**FsiEncrypt**

**Option bei FsiFileProcess(), durch die eine externe Datei entschlüsselt und dekomprimiert werden kann.**



**\_FsiDeleteOnClose**

**Datei nach dem Schließen löschen**

Wert 32.768 /

**0x00008000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die eine Datei temporär angelegt werden kann. Die Datei wird nach dem Schließen mit **FsiClose()** wieder gelöscht.

Die Option muss mit Optionen zum Lesen und Schreiben kombiniert werden.



Die Datei kann nach dem Öffnen von keinem anderen Programm mehr geöffnet werden.



Die Option kann nur unter Windows verwendet werden. Sie wird unter Linux ignoriert.

## Kontakt

**\_FsiDenyNone**

**Kein exklusiver Zugriff**

Wert **0** /

**0x00000000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die eine Datei ohne exklusiven Zugriff geöffnet werden kann.

Auf eine ohne exklusivem Zugriff geöffnete Datei sind Zugriffe anderer Benutzer oder Prozesse möglich.

**\_FsiDenyR**

**Exklusiver Lesezugriff**

Wert <sup>4/</sup>

**0x00000004**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die eine Datei mit exklusivem Lesezugriff geöffnet werden kann.

Auf eine mit exklusivem Lesezugriff geöffnete Datei sind keine weiteren Lesezugriffe anderer Benutzer oder Prozesse möglich.

Diese Option ist unter UNIX ohne Wirkung.

**\_FsiDenyRW**

**Exklusiver Lese- und Schreibzugriff**

Wert <sup>12/</sup>

**0x0000000C**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die eine Datei mit exklusivem Lese- und Schreibzugriff geöffnet werden kann.

Auf eine mit exklusivem Lese- und Schreibzugriff geöffnete Datei sind keine weiteren Lese- und Schreibzugriffe anderer Benutzer oder Prozesse möglich.

Diese Option ist unter UNIX ohne Wirkung.

## Kontakt

**\_FsiDenyW**

**Exklusiver Schreibzugriff**

Wert <sup>8/</sup>

**0x00000008**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die eine Datei mit exklusivem Schreibzugriff geöffnet werden kann.

Auf eine mit exklusivem Schreibzugriff geöffnete Datei sind keine weiteren Schreibzugriffe anderer Benutzer oder Prozesse möglich.

Diese Option ist unter UNIX ohne Wirkung.

**\_FsiDiskAvailMB**

Verfügbare Kapazität in MB ermitteln

Wert 36

**Verwandte**

Siehe **Befehle**,

**FsiDiskInfo()**

Option bei **FsiDiskInfo()** durch die die verfügbare Kapazität eines Datenträgers in MB ermittelt werden kann. Sind mehr als 2 PB verfügbar, wird **\_MaxInt** zurückgegeben.

Die verfügbare Kapazität wird durch Datenträgerkontingente beschränkt.

## Kontakt

**\_FsiDiskExists**

Datenträgerexistenz ermitteln

Wert 3

**Verwandte**

Siehe **Befehle**,

**FsiDiskInfo()**

Option bei **FsiDiskInfo()** durch die die Existenz eines Datenträgers ermittelt werden kann.

Falls der Datenträger existiert: Resultat = 1, ansonsten Resultat = 0.

\_FsiDiskFree

Freie Kapazität in KB ermitteln

Wert 0

### Verwandte

Siehe Befehle,

FsiDiskInfo(),

FsiDiskFreeMB

Option bei FsiDiskInfo() durch die die freie Kapazität eines Datenträgers in KB ermittelt werden kann. Sind mehr als 2 TB frei, wird \_MaxInt zurückgegeben. In diesem Fall sollte \_FsiDiskFreeMB verwendet werden.



## Kontakt

\_FsiDiskFreeMB

Freie Kapazität in MB ermitteln

Wert 32

Verwandte

Siehe Befehle,

FsiDiskInfo(),

FsiDiskFree

Option bei FsiDiskInfo() durch die die freie Kapazität eines Datenträgers in MB ermittelt werden kann. Sind mehr als 2 PB frei, wird MaxInt zurückgegeben.

## Kontakt

**\_FsiDiskReady**

Datenträgerbereitschaft ermitteln

Wert 2

**Verwandte**

Siehe **Befehle**,

**FsiDiskInfo()**

Option bei **FsiDiskInfo()** durch die die Bereitschaft eines Datenträgers ermittelt werden kann.

Falls der Datenträger bereit ist: Resultat = 1, ansonsten Resultat = 0.

\_FsiDiskTotal

Gesamte Kapazität in KB ermitteln

Wert 1

Verwandte

Siehe Befehle,

FsiDiskInfo(),

FsiDiskTotalMB

Option bei FsiDiskInfo() durch die die gesamte Kapazität eines Datenträgers in KB ermittelt werden kann. Sind mehr als 2 TB vorhanden, wird \_MaxInt zurückgegeben. In diesem Fall sollte FsiDiskTotalMB verwendet werden.

## Kontakt

**\_FsiDiskTotalMB**

Gesamte Kapazität in MB ermitteln

Wert 33

**Verwandte**

Siehe Befehle,

**FsiDiskInfo()**,

**FsiDiskTotal**

Option bei FsiDiskInfo() durch die die gesamte Kapazität eines Datenträgers in MB ermittelt werden kann. Sind mehr als 2 PB vorhanden, wird MaxInt zurückgegeben.

**\_FsiDtAccessed**

**Letzten Zugriffszeitpunkt ermitteln/setzen**

**Wert 0**

### **Verwandte**

Siehe **Befehle**,

**FsiDate()**,

**FsiTime()**

**Option bei FsiDate() und FsiTime() durch die das Datum bzw. die Uhrzeit des letzten Zugriffs ermittelt/gesetzt werden kann.**

**Diese Werte sind nicht bei FAT- und UNIX-Dateisystemen verfügbar.**

**\_FsiDtCreated**

Erstellungszeitpunkt ermitteln/setzen

Wert 2

### **Verwandte**

Siehe Befehle,

FsiDate(),

FsiTime()

Option bei FsiDate() und FsiTime() durch die das Datum bzw. die Uhrzeit der Erstellung ermittelt/gesetzt werden kann.

Diese Werte sind nicht bei FAT-Dateisystemen verfügbar.

## Kontakt

\_FsiDtModified

Letzten Änderungszeitpunkt ermitteln/setzen Wert 1

### Verwandte

Siehe Befehle,

FsiDate(),

FsiTime()

Option bei FsiDate() und FsiTime() durch die das Datum bzw. die Uhrzeit der letzten Änderung ermittelt/gesetzt werden kann.

Diese Werte sind bei allen Dateisystemen verfügbar.

**\_FsiEncrypt**

**Datei verschlüsseln**

**Wert 256 / 0x0100**

**Verwandte**

**Befehle,**

**Siehe FsiFileProcess(),**

**FsiFileMD5,**

**FsiDecode**

**Option bei FsiFileProcess(), durch die eine externe Datei verschlüsselt werden kann.**



\_FsiFileCRC32

CRC32-Prüfsumme der Datei

Wert 9

Verwandte

Befehle,

FsiFileInfo(),

FsiFileProcess(),

Siehe FsiFileRMD160,

FsiFileSHA1,

FsiFileSHA256,

FsiFileSHA384,

FsiFileSHA512

Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird die CRC32-Prüfsumme berechnet. Die Summe wird als Zeichenkette aus 8 Hexadezimalziffern zurückgegeben. Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.



Dieses Verfahren ist nicht kollisionssicher. Statt dessen sollte FsiFileSHA256,  
FsiFileSHA384 oder FsiFileSHA512 verwendet werden.

\_FsiFileMD5

MD-5 Hash der Datei

Wert 3

Verwandte

Befehle,

FsiFileInfo(),

FsiFileProcess(),

Siehe FsiFileRMD160,

FsiFileSHA1,

FsiFileSHA256,

FsiFileSHA384,

FsiFileSHA512

Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird mit der MD-5 Hash-Funktion eine Summe berechnet. Die Summe wird als Zeichenkette aus 32 Hexadezimalziffern zurückgegeben. Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.



Dieses Verfahren ist nicht kollisionssicher. Statt dessen sollte FsiFileSHA256,  
FsiFileSHA384 oder FsiFileSHA512 verwendet werden.

\_FsiFileRMD160

Hash der Datei nach RIPEMD-160-Standard Wert

4

Verwandte

Befehle,

FsiFileInfo(),

FsiFileProcess(),

Siehe FsiFileMD5,

FsiFileSHA1,

FsiFileSHA256,

FsiFileSHA384,

FsiFileSHA512

Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird ein Hash-Wert nach RIPEMD-160-Standard berechnet. Der Wert wird als Zeichenkette aus 40 Hexadezimalziffern zurückgegeben. Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.

Im Vergleich zum MD5-Hash-Wert dauert die Ermittlung des Hashwerts praktisch nicht länger, die Sicherheit ist jedoch höher, da im Gegensatz zu MD5 bisher keine Berechnung von Kollisionen möglich ist. Dadurch wird ein besserer Schutz vor Manipulationen der Dateiinhalte erreicht.

**\_FsiFileSHA1**

**SHA-1-Hash der Datei**

**Wert 5**

**Verwandte**

**Befehle,**

**FsiFileInfo(),**

**FsiFileProcess(),**

**Siehe FsiFileMD5,**

**FsiFileRMD160,**

**FsiFileSHA256,**

**FsiFileSHA384,**

**FsiFileSHA512**

**Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird ein Hash-Wert nach SHA-1-Standard berechnet. Der Wert wird als Zeichenkette aus 40 Hexadezimalziffern zurückgegeben. Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.**

**\_FsiFileSHA256**

**SHA-256-Hash der Datei**

**Wert 6**

**Verwandte**

**Befehle,**

**FsiFileInfo(),**

**FsiFileProcess(),**

**Siehe FsiFileMD5,**

**FsiFileRMD160,**

**FsiFileSHA1,**

**FsiFileSHA384,**

**FsiFileSHA512**

**Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird ein 256-Bit-Hash-Wert nach SHA-2-Standard berechnet. Der Wert wird als Zeichenkette aus 64 Hexadezimalziffern zurückgegeben.**

**Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.**

**\_FsiFileSHA384**

**SHA-384-Hash der Datei**

**Wert 7**

**Verwandte**

**Befehle,**

**FsiFileInfo(),**

**FsiFileProcess(),**

**Siehe FsiFileMD5,**

**FsiFileRMD160,**

**FsiFileSHA1,**

**FsiFileSHA256,**

**FsiFileSHA512**

**Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird ein 384-Bit-Hash-Wert nach SHA-2-Standard berechnet. Der Wert wird als Zeichenkette aus 96 Hexadezimalziffern zurückgegeben.**

**Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.**

\_FsiFileSHA512

SHA-512-Hash der Datei

Wert 8

Verwandte

Befehle,

FsiFileInfo(),

FsiFileProcess(),

Siehe FsiFileMD5,

FsiFileRMD160,

FsiFileSHA1,

FsiFileSHA256,

FsiFileSHA384

Option bei FsiFileInfo() und FsiFileProcess(). Aus der Datei wird ein 512-Bit-Hash-Wert nach SHA-2-Standard berechnet. Der Wert wird als Zeichenkette aus 128 Hexadezimalziffern zurückgegeben. Über diese Funktion kann bestimmt werden, ob Dateien verändert wurden.

## Kontakt

**\_FsiFileVersion**

Version der Datei ermitteln

Wert 1

**Verwandte**

Siehe **Befehle**,

**FsiFileInfo()**

Option bei **FsiFileInfo()**. Wird diese Option angegeben, wird die Version der Datei zurückgegeben. Die einzelnen Zahlen der Version sind durch Punkte voneinander getrennt. Unter Linux-Systemen wird eine leere Zeichenkette zurückgegeben.



## Kontakt

**\_FsiFileVersionHex**

Versionsnummer der Datei im Hexadezimalformat Wert 2

### **Verwandte**

Siehe **Befehle**,

### **FsiFileInfo()**

Option bei **FsiFileInfo()**. Wird diese Option angegeben besteht der Rückgabewert aus 16 Hexadezimalzahlen. Jeweils vier Ziffern bilden einen Teil der Versionsnummer. Die einzelnen Blöcke sind nicht voneinander getrennt. Unter Linux-Systemen wird eine leere Zeichenkette zurückgegeben.

**\_FsiGroupR**

**Leseberechtigung für Gruppe**

Wert 262.144 /

**0x00040000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die einer Datei beim Anlegen eine Leseberechtigung für die Benutzergruppen zugeteilt werden kann, sofern die gesetzte UMASK dies erlaubt.

Diese Option ist nur unter UNIX-Systemen relevant.

## Kontakt

**\_FsiGroupW**

**Schreibberechtigung für Gruppe**

Wert 524.288 /

**0x00080000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die einer Datei beim Anlegen eine Schreibberechtigung für die Benutzergruppen zugeteilt werden kann, sofern die gesetzte UMASK dies erlaubt.

Diese Option ist nur unter UNIX-Systemen relevant.

## Kontakt

**\_FsiMonActionCreate**

**Datei angelegt**

**Wert 1**

**Siehe EvtFsiMonitor**

**Übergabeparameter bei dem Ereignis EvtFsiMonitor. In dem überwachten Verzeichnis wurde eine Datei angelegt.**

## Kontakt

**\_FsiMonActionDelete**

**Datei wurde gelöscht**

**Wert 2**

**Siehe EvtFsiMonitor**

**Übergabeparameter bei dem Ereignis EvtFsiMonitor. Eine Datei im überwachten Verzeichnis wurde gelöscht.**

## Kontakt

**\_FsiMonActionModify**

**Datei wurde verändert**

**Wert 3**

**Siehe EvtFsiMonitor**

**Übergabeparameter bei dem Ereignis EvtFsiMonitor. Eine Datei im überwachten Verzeichnis wurde verändert.**

## Kontakt

**\_FsiMonActionRename**

**Datei wurde umbenannt**

**Wert 4**

**Siehe EvtFsiMonitor**

**Übergabeparameter bei dem Ereignis EvtFsiMonitor. Eine Datei im überwachten Verzeichnis wurde umbenannt.**

## Kontakt

**\_FsiMonFlagsSubDirs**

**Untergeordnete Verzeichnisse mitüberwachen Wert 1 /  
0x0001**

**Siehe EvtFsiMonitor,**

**Übergabeparameter bei dem Befehl FsiMonitorAdd(). Neben dem angegebenen Verzeichnis werden auch alle untergeordneten Verzeichnisse überwacht.**



## Kontakt

**\_FsiMonitorStart**

Verzeichnisüberwachung starten

Wert 1

Siehe **FsiMonitorControl()**

Übergabeparameter bei dem Befehl **FsiMonitorControl()**. Die Überwachung des Verzeichnisses wird gestartet.

## Kontakt

**\_FsiMonitorStop**

**Verzeichnisüberwachung anhalten**

**Wert 2**

**Siehe FsiMonitorControl()**

**Übergabeparameter bei dem Befehl FsiMonitorControl(). Die Überwachung des Verzeichnisses wird ausgesetzt.**

**\_FsiNameC16**

**Dateiname im CONZEPT 16-Zeichensatz**

**Wert 2.048 /**

**0x00000800**

**Verwandte**

**Befehle,**

**Siehe FsiOpen(),**

**FsiDirOpen(),**

**FsiAttributes(),**

**FsiNameUtf8**

**Option bei FsiOpen(), FsiDirOpen() und FsiAttributes(), mit der Dateinamen im CONZEPT 16-Zeichensatz interpretiert werden. Die Option kann nicht mit FsiNameUtf8 kombiniert werden.**

## Kontakt

**\_FsiNameE**

**Erweiterung ermitteln**

**Wert 6 / 0x06**

**Verwandte**

**Siehe Befehle,**

**FsiSplitName()**

**Option bei FsiSplitName() durch die aus einem Dateiname/-pfad die Erweiterung ermittelt werden kann.**

**Beispiel:**

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNameE) // 'ASC'FsiSplitName('INFO.ASC', _FsiNameE)
```

## Kontakt

\_FsiNameN

Datei ermitteln

Wert 5 / 0x05

Verwandte

Siehe Befehle,

FsiSplitName()

Option bei FsiSplitName() durch die aus einem Dateiname/-pfad die Datei ermittelt werden kann.

**Beispiel:**

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNameN) // 'INFO'FsiSplitName('INFO.ASC', _FsiNameN)
```

## Kontakt

\_FsiNameNE

Datei und Erweiterung ermitteln

Wert 4 / 0x04

Verwandte

Siehe Befehle,

FsiSplitName()

Option bei FsiSplitName() durch die aus einem Dateiname/-pfad die Datei und die Erweiterung ermittelt werden können.

Beispiel:

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNameNE) // 'INFO.ASC' FsiSplitName('INFO.ASC', _FsiNameNE)
```

## Kontakt

\_FsiNameP

Laufwerk und Pfad mit Separator ermitteln

Wert 3 / 0x03

Verwandte

Siehe Befehle,

FsiSplitName()

Option bei FsiSplitName() durch die aus einem Dateiname/-pfad das Laufwerk und der Pfad mit Separator ermittelt werden können.

Beispiele:

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNameP) // 'C:\TEST\FsiSplitName('INFO.ASC', _FsiNameP)
```

## Kontakt

**\_FsiNamePN**

**Laufwerk, Pfad und Datei ermitteln**

**Wert 1 / 0x01**

**Verwandte**

**Siehe Befehle,**

**FsiSplitName()**

**Option bei FsiSplitName() durch die aus einem Dateiname/-pfad das Laufwerk, der Pfad und die Datei ermittelt werden können.**

**Beispiel:**

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNamePN) // 'C:\TEST\INFO'FsiSplitName('INFO.ASC', _FsiNamePN
```



## Kontakt

\_FsiNamePNE

Laufwerk, Pfad, Datei und Erweiterung ermitteln Wert 0 / 0x00

### Verwandte

Siehe Befehle,

### FsiSplitName()

Option bei FsiSplitName() durch die aus einem Dateiname/-pfad das Laufwerk, der Pfad, die Datei und die Erweiterung ermittelt werden können.

Der übergebene Dateiname/-pfad bleibt unverändert.

**Beispiel:**

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNamePNE) // 'C:\TEST\INFO.ASC'FsiSplitName('INFO.ASC', _FsiN
```

## Kontakt

**\_FsiNamePP**

**Laufwerk und Pfad ohne Separator ermitteln**

**Wert 11 / 0x0B**

**Verwandte**

**Siehe Befehle,**

**FsiSplitName()**

**Option bei FsiSplitName() durch die aus einem Dateiname/-pfad das Laufwerk und der Pfad ohne Separator ermittelt werden können.**

**Beispiel:**

```
FsiSplitName('C:\TEST\INFO.ASC', _FsiNamePP) // 'C:\TEST'FsiSplitName('INFO.ASC', _FsiNamePP)
```

**\_FsiNameUtf8**

**Dateiname im UTF-8-Zeichensatz**

Wert 536.870.912 /

**0x20000000**

**Verwandte**

**Befehle,**

Siehe **FsiOpen()**,

**FsiDirOpen()**,

**FsiAttributes()**,

**FsiNameC16**

Option bei **FsiOpen()**, **FsiDirOpen()** und **FsiAttributes()**, mit der Dateinamen im UTF-8-Zeichensatz interpretiert werden. Die Option kann nicht mit **FsiNameC16** kombiniert werden.

## Kontakt

**\_FsiNoCache**

**Ungepuffertes Lesen/Schreiben**

Wert <sup>256</sup> /

**0x00000100**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die ein Zugriff auf den Cache beim Lesen bzw. Schreiben umgangen werden kann.

**\_FsiOtherR**

**Leseberechtigung für Andere**

Wert 1.048.576 /

**0x00100000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die einer Datei beim Anlegen eine Leseberechtigung für andere Benutzer zugeteilt werden kann, sofern die gesetzte UMASK dies erlaubt.

Diese Option ist nur unter UNIX-Systemen relevant.

## Kontakt

**\_FsiOtherW**

**Schreibberechtigung für Andere**

Wert 2.097.152 /

**0x00200000**

**Verwandte**

**Siehe Befehle,**

**FsiOpen()**

**Option bei FsiOpen() durch die einer Datei beim Anlegen eine Schreibberechtigung für andere Benutzer zugeteilt werden kann, sofern die gesetzte UMASK dies erlaubt.**

**Diese Option ist nur unter UNIX-Systemen relevant.**

\_FsiPure

Keine Zeichenwandlung beim Lesen oder Schreiben einer Datei **Wert**  
134.217.728

/ 0x08000000

Verwandte

Befehle,

Siehe FsiOpen(),

FsiRead(),

FsiWrite()

Option bei FsiOpen(). Standardmäßig erfolgt eine Umwandlung zwischen der internen und der OEM-Zeichencodierung beim Lesen (siehe FsiRead()) bzw. Schreiben (siehe FsiWrite()) von bzw. in eine externe Datei. Durch die Angabe dieser Konstanten wird keine Umwandlung durchgeführt, sodass bereits konvertierte Zeichenketten geschrieben werden können. Die Option kann nicht mit FsiANSI kombiniert werden.

**\_FsiStdRead**

**Standard-Lesemodus**

Wert <sup>521</sup> /

**0x00000209**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die eine Datei im Standard-Lesemodus geöffnet werden kann.

Diese Option entspricht folgender Zugriffsoption:

**\_FsiAcsR** | **\_FsiDenyW**



\_FsiStdWrite

Standard-Schreibmodus

Wert <sup>574</sup>/

0x0000023E

Verwandte

Siehe Befehle,

FsiOpen()

Option bei FsiOpen() durch die eine Datei im Standard-Schreibmodus geöffnet werden kann.

Diese Option entspricht folgender Zugriffsoption:

\_FsiAcsW | \_FsiDenyRW | \_FsiCreate | \_FsiTruncate

**\_FsiSyncWrite**

**Synchrones Schreiben**

Wert <sup>128 /</sup>

**0x00000080**

**Verwandte**

**Siehe Befehle,**

**FsiOpen()**

Option bei FsiOpen() durch die nachfolgende Schreiboperationen synchron ausgeführt werden können. Das bedeutet, dass die Schreiboperation erst dann beendet ist, wenn die Daten physikalisch auf dem Datenträger geschrieben wurden. Dies kann Schreiboperationen erheblich verlangsamen.

Diese Option ist nicht bei allen Betriebssystemen wirksam. Insbesondere in Netzwerkumgebungen wird ein synchrones Schreiben nicht immer unterstützt.

Synchrones Schreiben ist zur Zeit nur unter Windows-Systemen möglich.

**\_FsiTruncate**

**Datei leeren**

Wert <sup>32</sup> /

**0x00000020**

**Verwandte**

Siehe Befehle,

FsiOpen(),

FsiAppend

Option bei FsiOpen() durch die eine Datei auf die Größe 0 Byte geleert werden kann.  
Die Datei muss bereits existieren, ansonsten wird diese Option ignoriert.

Soll eine neue Datei angelegt werden, muss die Option \_FsiCreate verwendet werden.

**\_FsiUserR**

**Leseberechtigung für Benutzer**

Wert 65.536 /

**0x00010000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die einer Datei beim Anlegen eine Leseberechtigung für den Benutzer zugeteilt werden kann, sofern die gesetzte UMASK dies erlaubt.

Diese Option ist nur unter UNIX-Systemen relevant.

**\_FsiUserW**

**Schreibberechtigung für Benutzer**

Wert 131.072 /

**0x00020000**

**Verwandte**

Siehe **Befehle**,

**FsiOpen()**

Option bei **FsiOpen()** durch die einer Datei beim Anlegen eine Schreibberechtigung für den Benutzer zugeteilt werden kann, sofern die gesetzte UMASK dies erlaubt.

Diese Option ist nur unter UNIX-Systemen relevant.

## Systemfunktionen

### Befehle für Systemfunktionen

Siehe Befehlsgruppen Befehlsliste,

### Befehle

- ProcCacheClear
- ProcCompile
- RmtDataRead
- RmtDataSearch
- RmtDataWrite
- SysBeep
- SysClone
- SysDate
- SysExecute
- SysGetArg
- SysGetEnv
- SysOS
- SysSleep
- SysTics
- SysTime
- SysTimerClose
- SysTimerCreate

### Konstanten

- CloneAdvanced
- CloneMaximized
- CloneMinimized
- CloneStandard
- ExecHidden
- ExecMaximized
- ExecMinimized
- ExecWait
- RmtDataTemp
- TimeHSec
- TimeSec
- TimeServer

**ProcCompile(alpha1) : int**  **Prozedur**  
zur Laufzeit übersetzen

alpha1

Name der

**Prozedur**  
**Resultat int Fehlerwert**  —  
**ProcCacheClear(),**

Siehe

**System**

Mit dieser Anweisung wird die in (alpha1) übergebene Prozedur übersetzt. Als Resultat wird der Fehler beim Übersetzen zurückgegeben. Bei der Rückgabe von ErrOk konnte die Prozedur übersetzt werden. Der Fehlerwert steht ebenfalls in der Eigenschaft ErrCode zur Verfügung. Er kann mit einer der ErrCpl...-Konstanten verglichen werden.

Ist bei der Übersetzung ein Fehler aufgetreten, können weitere Informationen über folgende Eigenschaften des System-Objekts ermittelt werden:

<u>ErrCode</u>	Fehlerwert
<u>ErrText</u>	Fehlertext
<u>ErrProc</u>	Name der übersetzten Prozedur
<u>ErrSource</u>	Name der Quelltextprozedur (kann vom Namen der übersetzten Prozedur abweichen, wenn eine Prozedur mit der Include- <u>Anweisung</u> eingebunden wird)
<u>ErrSourceLine</u>	Quelltextzeile in der der Fehler aufgetreten ist
<u>ErrLine</u>	Nummer der Fehlerzeile im Quelltext
<u>ErrPos</u>	Fehlerposition in der Zeile

Unter folgenden Umständen kann eine Prozedur nicht übersetzt werden:

- Die Prozedur enthält ein with-Konstrukt.
- Die Prozedur wird gerade ausgeführt oder ist von einem anderen Client gesperrt.
- Die Prozedur ist keine A+ Prozedur (siehe @A+).

Falls die übersetzte Prozedur globale Datenbereiche (siehe global) enthält, dürfen diese nicht verändert werden, da die Änderungen nicht in die Runtime-Umgebung übernommen werden können und somit die Gefahr von Laufzeitfehlern bei Prozeduren, die diese Datenbereiche verwenden entsteht.



Falls Elemente der Prozedur geändert werden, die von anderen Prozeduren verwendet werden (andere Argumente bei Funktionen, Modifikationen von globalen Datenbereichen oder ähnliches) kann dies zu Laufzeitfehlern oder im ungünstigsten Fall zum Beenden des Clients führen. Dies gilt auch für alle anderen aktiven Clients der Datenbank, sofern diese die übersetzte Prozedur verwenden.

Sofern die übersetzte Prozedur auf anderen Arbeitsstationen aufgerufen wird, müssen in diesen Clients eventuell vorhandene ältere Codefragmente der übersetzten Prozedur aus dem Prozedurcache entfernt werden. Andernfalls können Laufzeitfehler entstehen. Dies wird mit dem Befehl ProcCacheClear() durchgeführt.

**ProcCacheClear(alpha1)**  **Prozedur**  
aus Cache entfernen

alpha1 Name der  
Prozedur

Siehe ProcCompile()

**Diese Anweisung entfernt die in (alpha1) übergebene Prozedur aus dem Prozedurcache des Clients. Die Funktion muss aufgerufen werden, wenn die Prozedur von einem anderen Client neu übersetzt wurde. Wird die Prozedur nicht aus dem Cache entfernt, werden möglicherweise Fragmente aus der alten Prozedur ausgeführt, was zu Laufzeitfehlern oder im ungünstigsten Fall zum Beenden des Clients führen kann.**



SysBeep(int1, int2) 

Akkustiksignal generieren

int1    Signalfrequenz    Hertz    in

int2

Signaldauer in

Millisekunden

Verwandte

Befehle,

Siehe WinBeep(),

Benutzerdefinierte

Sounds (Blog)

Mit diesem Befehl kann ein akustisches Signal erzeugt werden.


Beispiel:

```
SysBeep(440, 500) // Erzeugt eine halbe Sekunde lang einen 440 Hz Ton.
```



Wird der Befehl vom Server durchgeführt, kann abhängig von der Plattform die Tonhöhe und -dauer nicht beeinflusst werden.

## Kontakt

SysClone(int1[, alpha2[,  
alpha3[, alpha4[, alpha5]]]]) :   
int

Neuen Client-Prozess starten

	Optionen	
	<u>_CloneMinimized</u>	Neuen Client minimiert starten
int1	<u>_CloneMaximized</u>	Neuen Client maximiert starten
	<u>_CloneStandard</u>	Standard-Client starten
	<u>_CloneAdvanced</u>	Advanced-Client starten
alpha2		abweichende Startprozedur
alpha3		Kommandozeilenargumente
alpha4		abweichender Benutzer
alpha5		Passwort für Benutzer

**Resultat** int Prozess-ID oder  
Fehlerwert 

Siehe Verwandte Befehle

Mit diesem Befehl wird ein neuer Client-Prozess gestartet.

Folgende Optionen (int1) können angegeben werden:

<u>_CloneMinimized</u>	Der neue Client wird minimiert gestartet.
<u>_CloneMaximized</u>	Der neue Client wird maximiert gestartet.
<u>_CloneStandard</u>	Es wird der <u>Standard-Client</u> gestartet. Ist die Datenbank auf Stand 3.x wird der Client für 3.x-Datenbanken verwendet.
<u>_CloneAdvanced</u>	Es wird der <u>Advanced-Client</u> gestartet.

Die Optionen zur Größe ( \_CloneMinimized und \_CloneMaximized) können mit den Optionen zum Clienttyp ( \_CloneStandard und \_CloneAdvanced) kombiniert werden, jedoch nicht untereinander.

Ist keine der Optionen \_CloneStandard und \_CloneAdvanced angegeben, wird der aktuelle Clienttyp gestartet.

Es kann eine abweichende Startprozedur (alpha2) angegeben werden. Ist diese nicht angegeben, wird die Startprozedur ausgeführt, die in der Benutzerverwaltung für den Benutzer hinterlegt ist.

In dem Parameter (alpha3) können Kommandozeilenargumente angegeben werden.  
Bisherige Argumente können mit SysGetArg() ermittelt werden.

Wird ein Benutzer (alpha4) angegeben, wird dieser zum Start des Clients verwendet. In diesem Fall muss, sofern für den Benutzer ein Passwort definiert ist, sein Passwort (alpha5) angegeben werden. Ist kein Benutzer angegeben, wird der aktuelle Benutzer

## Kontakt

verwendet.

Im Fehlerfall wird ErrGeneric zurückgegeben, andernfalls die Prozess-ID des gestarteten Prozesses.

Beispiele:

```
// Neuen Client abhängig vom aktuellen Clienttyp startenSysClone(0); // Standard-Client unabhängig
```

**SysDate() : date**

Systemdatum ermitteln

Resultat date Systemdatum

Verwandte

Befehle,

Siehe Datumsfunktionen,

SysTime(),

SysTics(), caltime

Mit diesem Befehl kann das aktuelle Systemdatum ermittelt werden.



Die Anweisung ist nur noch aus Kompatibilitätsgründen im Sprachumfang enthalten. Es sollte die Methode ymSystemTime() oder ymServerTime() verwendet werden.

**SysExecute(alpha1,  
alpha2, int3) : int**



**Programm starten**

**alpha1** Programmname und -pfad

**alpha2** Programmparameter

**Optionen**

ExecWait Synchroner  
Programmverarbeitung

ExecMaximized Programm maximiert

**int3** starten

ExecMinimized Programm minimiert

ExecHidden starten  
Programm versteckt

starten

**Startresultat**

Programmrückgabewert (in  
Verbindung mit ExecWait)

**Resultat int** ErrOk Starten erfolgreich  
ErrGeneric Programm (alpha1)  
kann nicht  
gefunden/gestartet  
werden.

**Siehe** Verwandte Befehle

Mit diesem Befehl wird ein Programm gestartet, in (alpha1) wird der Pfad und der Name des Programms angegeben und in (alpha2) die Parameter, die an das aufzurufende Programm übergeben werden sollen. Sollen keine Parameter übergeben werden, kann in (alpha2) '' übergeben werden.



Die Anweisung kann von der DLL-Schnittstelle, jedoch nicht von der PHP-Schnittstelle ausgeführt werden.

**Beispiel:**

```
// Starten des Windows-Taschenrechners SysExecute('Calc', '', 0);
```

Über SysExecute() ist auch das direkte Öffnen von registrierten Dateitypen möglich. In diesem Fall wird in (alpha1) ein Stern (\*) gefolgt vom Namen des Dokumentes angegeben.

**Beispiel:**

```
// Starten eines Word-Dokuments SysExecute('*D:\Doc\ReadMe.doc', '', 0);
```

Um Befehle der Shell des Betriebssystems ausführen zu können, muss in (alpha1) der Windows-Befehlsprozessor durch die Angabe von 'cmd' gestartet werden. In (alpha2) wird der Shell-Befehl inklusive Parameter angegeben. Damit der Shell-Befehl auch ausgeführt wird, ist in (alpha2) zusätzlich die Angabe der cmd-Parameter '/c' oder '/k' (Optionen des Befehlsprozessors) notwendig. Eine Übersicht der Parameter des Befehlsprozessors kann mit "help cmd" in der Kommandozeile abgerufen werden.

### Beispiel:

```
// Ausführen des Shell-Befehls copySysExecute('cmd', '/c copy ' + _Sys->spPathTemp + '\a.dat ' +
```

In (int3) können folgende symbolische Konstanten übergeben werden:

- ExecWait

Das Programm wird solange angehalten, bis das aufgerufene Programm beendet wurde.  
Ohne die Angabe dieses Parameters werden beide Programme fortgesetzt.

- ExecMaximized


Das Programm wird maximiert gestartet.

- ExecMinimized

Das Programm wird minimiert gestartet.

- ExecHidden

Das Programm wird versteckt gestartet.

**SysGetArg(alpha1) : alpha** 

**Kommandozeilenargumente ermitteln**

**alpha1** Argumentname

**Resultat** alpha Argumentwert

Verwandte Befehle,

Siehe

SysGetEnv(), Blog

Auf der Kommandozeile bzw. in den Startparametern von CONZEPT 16 können anwendungsspezifische Argumente in folgender Form angegeben werden:

`/<Argumentname>=<Argumentwert>`

Mit SysGetArg() kann dann der Argumentwert des Arguments (alpha1) ermittelt werden. Ist kein Argument mit diesem Namen vorhanden, wird ein leeres Resultat geliefert.

Die Argumente c16, c16cfg, c16tmp, c16lang und c16splashon (siehe FAQ) lassen sich mit dieser Funktion ebenfalls ermitteln.

**Beispiel:**

```
// CONZEPT 16 wurde mit dem Argument '/UserMode=Pflege' aufgerufen.tArg # SysGetArg('UserMode');
```

SysGetEnv(alpha1) : alpha        Werte von \_\_\_\_\_

Umgebungsvariablen ermitteln

alpha1 Variablenname

Resultat alpha Variablenwert

Verwandte Befehle,

Dateipfade mit

Siehe

Umgebungsvariablen

(Blog)

Mit dieser Funktion wird der Wert einer Umgebungsvariable ermittelt. Das Resultat ist leer, wenn kein Eintrag mit dem Namen (alpha1) vorhanden ist. Einige der Systemvariablen können auch über Eigenschaften des System-Objekts ermittelt werden.

**Beispiel:**

```
SysGetEnv('TEMP') // Liefert das Temporärverzeichnis zurück.
```



SysOS([logic1]) : alpha



Betriebssystem ermitteln

Buildnummer ermitteln

logic1 (optional)

Resultat alpha Betriebssystemname  
(siehe Text)

Siehe Verwandte Befehle

Dieser Befehl ermittelt den Namen des aktuellen Betriebssystems.

Als Resultat können folgende Werte zurückgegeben werden:

```
'Windows XP' 'Windows XP (64-Bit)' 'Windows Server 2003' 'Windows Server 2003 (64-Bit)' 'Windows Serv
```

Wird im optionalen Argument (logic1) true angegeben, wird unter Windows zusätzlich die Buildnummer ermittelt. Das Resultat kann beispielsweise wie folgt aussehen:

```
'Windows 10 (Build: 16299)' 'Windows 10 (64-Bit, Build: 16299)'
```



Wird die Funktion SysOS() über RmtCall() auf einem Linux-Server ausgeführt, besteht das Resultat aus Informationen über den Linux-Kernel.

Beispiel

```
'Linux kernel 5.2 (64-Bit)'
```

**SysSleep(int1)**

**Verarbeitung anhalten**

**Anhaltedauer**

**int1 in**

**Millisekunden**

**Verwandte**

**Siehe Befehle,**

**WinSleep()**

**Diese Funktion hält die Verarbeitung in einer Prozedur für eine bestimmte Dauer (int1) in Millisekunden an.**

**Hier kann ebenfalls eine Wartezeit von 0 Millisekunden angegeben werden. Mit diesem Kommando wird die aktuelle Zeitscheibe des Prozesses freigegeben und kann durch das Betriebssystem an einen anderen Prozess vergeben werden.**

**SysTics() : int**

**Systemlaufzeit ermitteln**

Systemlaufzeit in

**Resultat int Millisekunden**

Siehe **vmSystemTime()**,

**vmServerTime()**

Mit diesem Befehl kann unter Windows-Betriebssystemen die Zeit seit dem Systemstart in Millisekunden ermittelt werden. Unter Linux ist der ermittelte Wert unabhängig vom Systemstart, dafür abhängig von Datum und Uhrzeit.

Der Wert steht nicht in Beziehung zur Uhrzeit. Mit dem Resultat von SysTics() kann ein Zeitintervall relativ exakt bestimmt werden. Es ist zu beachten, dass auf Windows Rechnern die Zeitauflösung ca. 10 Millisekunden beträgt.

Der Wert wird vom Betriebssystem ohne Vorzeichen zurückgeben, in CONZEPT 16 aber mit Vorzeichen interpretiert. Zur Vereinfachung von Berechnungen sind nur positive Werte zulässig. Läuft das System länger als ca. 24 Tage 20 Stunden, fängt der Wert wieder bei 0 an.



Um eine genauere Zeitauflösung zu erhalten sollte ein 64-Bit-Zeitstempel mit **vmSystemTime()** ermittelt werden.

**Beispiel:**

```
tTimeTics # SysTics();
```

**Berechnung des Zeitabstandes zwischen zwei mit SysTics() ermittelten Werten.**

```
define( // Differenz mit Beachtung des Übertrags Sys.TicsDiff(aTicsBegin, aTicsEnd) : (int(aTic
```



**SysTime(int1) : time**  
Systemzeit ermitteln

**Optionen**  
TimeSec      Zeit mit Sekunden  
                     ermitteln  
TimeHSec   Zeit mit  
 int1           Hundertstelsekunden  
                     ermitteln  
TimeServer Systemzeit des  
                     Servers ermitteln

**Resultat** time      Systemzeit

**Verwandte Befehle,**

Siehe Zeitfunktionen, SysDate(),  
SysTics(), calttime

Mit diesem Befehl kann die aktuelle Systemzeit ermittelt werden.



Die Anweisung ist nur noch aus Kompatibilitätsgründen im Sprachumfang enthalten. Um die aktuelle Uhrzeit zu ermitteln sollte die Methode vmSystemTime() oder vmServerTime() verwendet werden.

Mit der Option TimeServer kann statt der Uhrzeit des eigenen Rechners die Uhrzeit des Servers ermittelt werden. Die Systemzeit des eigenen Rechners bleibt dabei unverändert. Diese Option kann mit TimeSec und TimeHSec kombiniert werden.

**Beispiele:**

```
SysTime(_TimeSec);SysTime(_TimeServer | _TimeSec);
```

obj -> SysTimerClose()  Zeitgesteuertes

Ereignis beenden

objTimer-Deskriptor

Verwandte

Siehe Befehle,

SysTimerCreate(),

EvtTimer

Mit dieser Funktion wird ein mit SysTimerCreate() erzeugter Timer beendet. Der Timer-Deskriptor ist danach nicht mehr gültig.

Mögliche Laufzeitfehler:

ErrHdlInvalid Timer-Deskriptor (obj) ungültig

## Kontakt

**SysTimerCreate(int1, int2[, handle3])**



**: handle** Zeitgesteuertes Ereignis  
starten

**int1** Zeitintervall in

Millisekunden

**int2** Wiederholungsanzahl  
Zielfenster-Deskriptor

**handle3 (optional)**

### Verwandte Befehle,

Siehe SysTimerClose(),  
EvtTimer

Mit dieser Funktion wird in einem bestimmten Intervall (int1) in Millisekunden das Ereignis EvtTimer ausgelöst. Der Minimalwert von (int1) beträgt 100 ms. Bei Angabe von -1 in (int2) wird das Ereignis unbegrenzt wiederholt, ansonsten enthält (int2) die Anzahl der auszulösenden Ereignisse. Bei (int2) gleich 0 wird kein Ereignis ausgelöst.

In (handle3) kann der Deskriptor des Frame-Objekts angegeben werden, welches das Ereignis erhält. Wird (handle3) nicht angegeben oder auf 0 gesetzt, erhalten alle Top-Level-Frames (die Frames ohne Parent) das Ereignis. Eine entsprechende Prozedurfunktion für das Ereignis EvtTimer muss beim jeweiligen Frame angegeben werden.

Das Resultat ist der Deskriptor des Timers.

Das Zeitintervall beginnt erst nach der kompletten Verarbeitung von EvtTimer erneut.

Damit wird verhindert, dass während der Ereignisverarbeitung bereits ein weiteres Ereignis ausgelöst wird.

Die maximale Anzahl von Timern ist nicht beschränkt. Es ist zu beachten, dass ein Timer immer mit SysTimerClose() entfernt werden muss, auch wenn er keine Ereignisse mehr auslöst.

Beispiel:

```
// Ein Timer auf den Dialog $Frame, mit// einem Intervall von 300 ms und einer// unendlichen Wied
```

Konstanten für Systemfunktionen

Konstanten für Systemfunktionen

Siehe Systemfunktionen

- CloneAdvanced
- CloneMaximized
- CloneMinimized
- CloneStandard
- ExecHidden
- ExecMaximized
- ExecMinimized
- ExecWait
- RmtDataTemp
- TimeHSec
- TimeSec
- TimeServer

## Kontakt

\_CloneAdvanced

Neuen Client als Advanced-Client starten

Wert 2 / 0x0002

Verwandte

Siehe Befehle,

SysClone()

Option bei SysClone() durch die der Advanced-Client gestartet werden kann.



## Kontakt

**\_CloneMaximized**

Neuen Client maximiert starten

Wert 32 / 0x0020

**Verwandte**

Siehe Befehle,

SysClone(),

CloneMinimized

Option bei SysClone() durch die der Client mit maximierter Darstellung gestartet werden kann.

**\_CloneMinimized**

**Neuen Client minimiert starten**

**Wert 16 / 0x0010**

**Verwandte**

Siehe Befehle,

SysClone(),

CloneMaximized

**Option bei SysClone() durch die der Client mit minimierter Darstellung gestartet werden kann.**

## Kontakt

\_CloneStandard

Neuen Client als Standard-Client starten

Wert 1 / 0x0001

Verwandte

Siehe Befehle,

SysClone()

Option bei SysClone() durch die der Standard-Client gestartet werden kann.

**\_ExecHidden**

**Programm versteckt ausführen**

Wert <sup>6/</sup>

**0x00000006**

**Verwandte**

**Siehe Befehle,**

**SysExecute()**

**Option bei SysExecute() durch die das Programm versteckt ausgeführt werden kann.**



**Diese Option wird nicht von allen Programmen unterstützt, vornehmlich aber von Kommandozeilen-basierten Anwendungen.**

**\_ExecMaximized**

**Programm maximiert starten**

**Wert 4 / 0x00000004**

**Verwandte**

Siehe Befehle,

SysExecute(),

ExecMinimized

**Option bei SysExecute() durch die ein Programm mit maximierter Darstellung gestartet werden kann.**

**\_ExecMinimized**

**Programm minimiert starten**

**Wert 2 / 0x00000002**

**Verwandte**

Siehe **Befehle**,

**SysExecute()**,

**ExecMaximized**

**Option bei SysExecute() durch die ein Programm mit minimierter Darstellung gestartet werden kann.**

**\_ExecWait**

**Synchrone Programmverarbeitung**

Wert <sup>1/</sup>

**0x00000001**

**Verwandte**

Siehe **Befehle**,

**SysExecute()**

Option bei **SysExecute()** durch die die Verarbeitung der Prozedur bis zum Beenden des gestarteten Programms angehalten werden kann.

## Kontakt

### \_RmtDataTemp

Aufbewahrung des zentralen Datenobjekts bis zur Abmeldung

Wert 1.073.741.824 /

0x40000000

Siehe RmtDataWrite()

Die Option wird beim Befehl RmtDataWrite() angegeben. Die geschriebenen Daten bleiben erhalten, bis sich der Benutzer, der die Daten geschrieben hat, von der Datenbank abmeldet. Wird die Option nicht angegeben, bleiben die Daten erhalten, bis die Datenbank geschlossen wird.



**\_TimeHSec**

**Zeit mit Hundertstelsekunden ermitteln**

**Wert 2 / 0x02**

**Verwandte**

**Siehe Befehle,**

**SysTime()**

**Option bei SysTime() durch die die Systemzeit mit Hundertstelsekunden ermittelt werden kann.**

**\_TimeSec**

Zeit mit Sekunden ermitteln

Wert 1 / 0x01

**Verwandte**

Siehe **Befehle**,

**SysTime()**

Option bei **SysTime()** durch die die Systemzeit mit Sekunden ermittelt werden kann.

## Kontakt

**\_TimeServer**

Systemzeit des Servers ermitteln

Wert 4 / 0x04

**Verwandte**

Siehe **Befehle**,

**SysTime()**

Option bei **SysTime()** durch die die Systemzeit des Servers ermittelt werden kann.

Befehle für Systemobjekte

Befehle für Systemobjekte

Siehe Befehlsgruppen Befehlsliste,

Befehle

- HttpClose
- HttpGetData
- HttpOpen
- LocaleLoad
- LocaleSelect
- LocaleUnload
- SysPropGet
- SysPropSet

Konstanten

- LclLangCzech
- LclLangEnglish
- LclLangFrench
- LclLangGerman
- LclLangHungarian
- LclLangItalian
- LclLangNeutral
- LclLangPolish
- LclLangSlovak
- LclLangTurkish
- LclSubLangDefault
- LclSubLangEnglishAU
- LclSubLangEnglishCA
- LclSubLangEnglishIE
- LclSubLangEnglishNZ
- LclSubLangEnglishUK
- LclSubLangEnglishUS
- LclSubLangEnglishZA
- LclSubLangFrench
- LclSubLangFrenchBE
- LclSubLangFrenchCA
- LclSubLangFrenchCH
- LclSubLangFrenchLU
- LclSubLangFrenchMC
- LclSubLangGerman
- LclSubLangGermanAT
- LclSubLangGermanCH
- LclSubLangGermanLI
- LclSubLangGermanLU
- LclSubLangItalian
- LclSubLangItalianCH
- LclSubLangNeutral

## Kontakt

**obj -> SysPropGet(int1, var2[,int3]) :**  
**logic Systemobjekteigenschaft ermitteln**



**obj**        **Objekt**  
**int1**       **Eigenschaftskonstante**  
**var2**       **Eigenschaftswert**  
**int3**       **Position**  
**Resultat logic** **Ermittlungserfolg**  
              **Verwandte Befehle,**  
**Siehe**      **SysPropSet(), Liste der**  
              **Systemobjekt-Eigenschaften**

Dieser Befehl liest eine Eigenschaft eines Systemobjektes aus.

Als erster Parameter muss die Konstante der Eigenschaft übergeben werden. Die Konstanten setzen sich aus `_SysProp` und dem Namen der Eigenschaft zusammen.

Im zweiten Parameter wird die Variable übergeben, in die der Wert der Eigenschaft kopiert werden soll.

**Beispiel:**

```
local{ tHdlLocale      : handle; tDateFormat      : alpha;}...// Auslesen des DatumsformatestHdlLo
```

Das Kommando kann ebenfalls dazu verwendet werden, um zu ermitteln, ob ein bestimmtes Objekt eine Eigenschaft besitzt. Ist eine Eigenschaft nicht vorhanden, liefert der Befehl den Wert false zurück.



Alternativ kann die Eigenschaft auch wie folgt ausgelesen werden:

**Beispiel:**

```
local{ tHdlLocale      : handle; tDateFormat      : alpha;}...// Auslesen des Datumsformatest
```

Der optionale Parameter (int3) muss nur angegeben werden, wenn mehrere Werte einer Eigenschaft zugeordnet werden können. Entsprechende Hinweise befinden sich in den Beschreibungen der Eigenschaften.

**Beispiel:**

```
// Name des Sonntags ermittelntHdlLocale->SysPropGet(_SysPropLclDateDayN, aSunday, 7);// Alternat
```

## Kontakt

**obj -> SysPropSet(int1, var2[,  
int3]) : logic**



**Systemobjekteigenschaft setzen**

**obj**        **Objekt**

**int1**       **Eigenschaftskonstante**

**var2**       **Eigenschaftswert**

**int3**       **Position**

**Resultat** **logic** **Setzungserfolg**

**Verwandte Befehle,**

**Siehe**      **SysPropGet(), Liste der**  
             **Systemobjekt-Eigenschaften**

**Dieser Befehl setzt eine Eigenschaft eines Systemobjektes.**

**Als ersten Parameter muss die Konstante der Eigenschaft übergeben werden. Die Konstanten setzen sich aus \_SysProp und dem Namen der Eigenschaft zusammen.**

**Im zweiten Parameter wird der zu setzende Wert übergeben.**

**Beispiel:**

```
// Setzen des kurzen DatumsformatesHdlLocale # LocaleLoad(_LclLangGerman, _LclSubLangGerman);tHd
```



**Alternativ kann die Eigenschaft auch wie folgt gesetzt werden:**

**Beispiel:**

```
// Setzen des kurzen DatumsformatesHdlLocale # LocaleLoad(_LclLangGerman, _LclSubLangGerman
```

**Der optionale Parameter (int3) muss nur angegeben werden, wenn mehrere Werte einer Eigenschaft zugeordnet werden können. Entsprechende Hinweise befinden sich in den Beschreibungen der Eigenschaften.**

**Beispiel:**

```
// Name des Sonntags setzentHdlLocale->SysPropSet(_SysPropLclDateDayN, 'Sonntag', 7);// Alternati
```

**LocaleLoad(int1, int2) :**



**handle**

**Ländereinstellungen laden**

**int1      Regionale ID**

**int2      Regionale Sub-ID**

**Resultat handle Locale-Objekt-Deskriptor**

**Verwandte Befehle, Locale,**

**Siehe**

**LocaleSelect(), LocaleUnload()**

Dieser Befehl lädt die länderspezifischen Einstellungen des in (int1) und (int2) angegebenen Landes. Die Einstellungen können über ein Locale-Objekt abgefragt oder verändert werden. Der Befehl gibt den Deskriptor zu dem entsprechenden Locale-Objekt als Rückgabewert zurück.

Folgende Werte können in (int1) und (int2) übergeben werden:

- LclLangNeutral Länderunabhängige Einstellungen
  - ◆ LclSubLangNeutral Länderunabhängige Einstellungen
- LclLangCzech Tschechisch
- LclLangEnglish Englisch
  - ◆ LclSubLangEnglishUS USA
  - ◆ LclSubLangEnglishUK Großbritannien
  - ◆ LclSubLangEnglishAU Australien
  - ◆ LclSubLangEnglishCA Kanada
  - ◆ LclSubLangEnglishNZ Neuseeland
  - ◆ LclSubLangEnglishIE Irland
  - ◆ LclSubLangEnglishZA Südafrika
- LclLangFrench Französisch
  - ◆ LclSubLangFrench Frankreich
  - ◆ LclSubLangFrenchBE Belgien
  - ◆ LclSubLangFrenchCA Kanada
  - ◆ LclSubLangFrenchCH Schweiz
  - ◆ LclSubLangFrenchLU Luxemburg
  - ◆ LclSubLangFrenchMC Monaco
- LclLangGerman Deutsch
  - ◆ LclSubLangGerman Deutschland
  - ◆ LclSubLangGermanCH Schweiz
  - ◆ LclSubLangGermanAT Österreich
  - ◆ LclSubLangGermanLU Luxemburg
  - ◆ LclSubLangGermanLI Liechtenstein
- LclLangHungarian Ungarisch
- LclLangItalian Italienisch
  - ◆ LclSubLangItalian Italien
  - ◆ LclSubLangItalianCH Schweiz

## Kontakt

- LclLangPolish Polnisch
- LclLangSlovak Slowakisch
- LclLangTurkish Türkisch

Wird in (int1) LclLangNeutral und in (int2) LclSubLangNeutral angegeben, wird eine länderunabhängige Einstellung geladen.

Mit der Kombination LclLangNeutral und LclSubLangDefault kann die Einstellung des Windows-Benutzers geladen werden.

Durch die Übergabe falscher Kombinationen wird ein leeres Locale-Objekt geladen.

Sind verschiedene Ländereinstellungen geladen, kann mit dem Befehl LocaleSelect() zwischen diesen Einstellungen gewechselt werden. Der Befehl LocaleUnload() entfernt das Locale-Objekt wieder aus dem Speicher.



obj -> LocaleSelect() 

Ländereinstellungen wechseln

objLocale-Objekt-Deskriptor

Verwandte Befehle,

**Siehe**

Locale, LocaleLoad()

Mit diesem Befehl kann zwischen mehreren geladenen Ländereinstellungen gewechselt werden. Die Ländereinstellungen müssen zuvor mit dem Befehl LocaleLoad() geladen worden sein. In (obj) wird der Deskriptor des Locale-Objektes angegeben, das die neuen Ländereinstellungen enthält.

Dies verändert die Einstellungen für alle Dialoge und A+ Prozeduren. Sollen die entsprechenden Einstellungen nicht global verwendet werden, können sie in den Konvertierungsbefehlen oder beim Oberflächenobjekt angegeben werden. Ein LocaleSelect() entfällt in diesem Fall.

Sollen die Systemeinstellungen wieder hergestellt werden, muss das zur Zeit selektierte Locale mit dem Befehl LocaleUnload() entladen werden.

## Kontakt

obj -> LocaleUnload() 

Ländereinstellungen entladen

objLocale-Objekt-Deskriptor

Verwandte Befehle,

**Siehe** Locale, LocaleLoad()

Mit diesem Befehl wird das in (obj) übergebene Locale-Objekt aus dem Speicher entfernt.

Konstanten für Systemobjekte

Konstanten für Systemobjekte

Siehe Befehle für  
Systemobjekte

- LclLangCzech
- LclLangEnglish
- LclLangFrench
- LclLangGerman
- LclLangHungarian
- LclLangItalian
- LclLangNeutral
- LclLangPolish
- LclLangSlovak
- LclLangTurkish
- LclSubLangDefault
- LclSubLangEnglishAU
- LclSubLangEnglishCA
- LclSubLangEnglishIE
- LclSubLangEnglishNZ
- LclSubLangEnglishUK
- LclSubLangEnglishUS
- LclSubLangEnglishZA
- LclSubLangFrench
- LclSubLangFrenchBE
- LclSubLangFrenchCA
- LclSubLangFrenchCH
- LclSubLangFrenchLU
- LclSubLangFrenchMC
- LclSubLangGerman
- LclSubLangGermanAT
- LclSubLangGermanCH
- LclSubLangGermanLI
- LclSubLangGermanLU
- LclSubLangItalian
- LclSubLangItalianCH
- LclSubLangNeutral

## Kontakt

**\_LclLangCzech**

**Ländereinstellungen Tschechisch**

**Wert 5 / 0x05**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von Tschechien geladen werden können. Als regionale Sub-ID muss \_LclSubLangDefault angegeben werden.**

\_LclLangEnglish

Ländereinstellungen Englisch

Wert 9 / 0x09

Verwandte

Siehe Befehle,

LocaleLoad()

Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von englischsprachigen Ländern geladen werden können. Als regionale Sub-ID können folgende Konstanten angegeben werden:

\_LclSubLangDefault Entspricht der Option \_LclSubLangEnglishUS

\_LclSubLangEnglishUS Einstellungen für USA

\_LclSubLangEnglishUK Einstellungen für Großbritannien

\_LclSubLangEnglishAU Einstellungen für Australien

\_LclSubLangEnglishCA Einstellungen für Kanada

\_LclSubLangEnglishNZ Einstellungen für Neuseeland

\_LclSubLangEnglishIE Einstellungen für Irland

\_LclSubLangEnglishZA Einstellungen für Südafrika

## Kontakt

\_LclLangFrench

Ländereinstellungen Französisch

Wert 12 / 0x0C

Verwandte

Siehe Befehle,

LocaleLoad()

Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von französischsprachigen Ländern geladen werden können. Als regionale Sub-ID können folgende Konstanten angegeben werden:

<u>_LclSubLangDefault</u>	Entspricht der Option <u>_LclSubLangFrench</u>
<u>_LclSubLangFrench</u>	Einstellungen für Frankreich
<u>_LclSubLangFrenchBE</u>	Einstellungen für Belgien
<u>_LclSubLangFrenchCA</u>	Einstellungen für Kanada
<u>_LclSubLangFrenchCH</u>	Einstellungen für Schweiz
<u>_LclSubLangFrenchLU</u>	Einstellungen für Luxemburg
<u>_LclSubLangFrenchMC</u>	Einstellungen für Monaco

## Kontakt

\_LclLangGerman

Ländereinstellungen Deutsch

Wert 7 / 0x07

Verwandte

Siehe Befehle,

LocaleLoad()

Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von deutschsprachigen Ländern geladen werden können. Als regionale Sub-ID können folgende Konstanten angegeben werden:

<u>_LclSubLangDefault</u>	Entspricht der Option <u>_LclSubLangGerman</u>
<u>_LclSubLangGerman</u>	Einstellungen für Deutschland
<u>_LclSubLangGermanCH</u>	Einstellungen für Schweiz
<u>_LclSubLangGermanAT</u>	Einstellungen für Österreich
<u>_LclSubLangGermanLU</u>	Einstellungen für Luxemburg
<u>_LclSubLangGermanLI</u>	Einstellungen für Liechtenstein

## Kontakt

**\_LclLangHungarian**

**Ländereinstellungen Ungarn**

**Wert 14 / 0x0E**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von Ungarn geladen werden können. Als regionale Sub-ID muss \_LclSubLangDefault angegeben werden.**



## Kontakt

\_LclLangItalian

Ländereinstellungen Italienisch

Wert 16 / 0x10

Verwandte

Siehe Befehle,

LocaleLoad()

Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von italienischsprachigen Ländern geladen werden können. Als regionale Sub-ID können folgende Konstanten angegeben werden:

\_LclSubLangDefault      Entspricht der Option \_LclSubLangItalian

\_LclSubLangItalian      Einstellungen für Italien

\_LclSubLangItalianCH Einstellungen für Schweiz

**\_LclLangNeutral**

**Länderunabhängige Einstellungen**

**Wert 0 / 0x00**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die eine länderunabhängige Einstellung geladen werden kann, wenn als regionale Sub-ID \_LclSubLangNeutral angegeben wird.**

**Mit der Angabe von \_LclSubLangDefault als regionale Sub-ID wird die Einstellung des Windows-Benutzers geladen.**

## Kontakt

**\_LclLangPolish**

**Ländereinstellungen Polen**

**Wert 21 / 0x15**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die die länderspezifischen Einstellungen von Polen geladen werden können. Als regionale Sub-ID muss \_LclSubLangDefault angegeben werden.**

## Kontakt

**\_LclLangSlovak**

**Ländereinstellungen Slowakei**

**Wert 27 / 0x1B**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die die länderspezifischen Einstellungen der Slowakei geladen werden können. Als regionale Sub-ID muss \_LclSubLangDefault angegeben werden.**

## Kontakt

**\_LclLangTurkish**

**Ländereinstellungen Türkei**

**Wert 31 / 0x1F**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die die länderspezifischen Einstellungen der Türkei geladen werden können. Als regionale Sub-ID muss \_LclSubLangDefault angegeben werden.**

**\_LclSubLangDefault**

**Regionaleinstellungen Standard**

**Wert 1 / 0x01**

**Verwandte**

**Siehe Befehle,**

**LocaleLoad()**

**Option bei LocaleLoad() durch die eine Standard-Regionaleinstellung geladen werden kann. Wurde als regionale ID LclLangNeutral angegeben, werden die länderspezifischen Einstellungen des Benutzers geladen.**

## Kontakt

**\_LclSubLangEnglishAU**

**Ländereinstellungen Australien (Englisch)**

**Wert 3 / 0x03**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangEnglish**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Australien (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangEnglish** angegeben wurde.

## Kontakt

\_LclSubLangEnglishCA

Ländereinstellungen Kanada (Englisch)

Wert 4 / 0x04

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangEnglish

Option bei LocaleLoad() durch die die Ländereinstellungen von Kanada (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID

LclLangEnglish angegeben wurde.



\_LclSubLangEnglishIE

Ländereinstellungen Irland (Englisch)

Wert 6 / 0x06

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangEnglish

Option bei LocaleLoad() durch die die Ländereinstellungen von Irland (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID

LclLangEnglish angegeben wurde.

## Kontakt

\_LclSubLangEnglishNZ Ländereinstellungen

Neuseeland (Englisch) Wert 5 / 0x05

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangEnglish

Option bei LocaleLoad() durch die die Ländereinstellungen von Neuseeland (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID LclLangEnglish angegeben wurde.

**\_LclSubLangEnglishUK**

**Ländereinstellungen Großbritannien (Englisch)**

**Wert 2 / 0x02**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangEnglish**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Großbritannien (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangEnglish** angegeben wurde.

\_LclSubLangEnglishUS

Ländereinstellungen USA (Englisch)

Wert 1 / 0x01

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangEnglish

Option bei LocaleLoad() durch die die Ländereinstellungen der USA (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID

LclLangEnglish angegeben wurde.

## Kontakt

**\_LclSubLangEnglishZA**

**Ländereinstellungen Südafrika (Englisch)**

**Wert 7 / 0x07**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangEnglish**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Südafrika (Englisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangEnglish** angegeben wurde.

**\_LclSubLangFrench**

**Ländereinstellungen Frankreich (Französisch)**

**Wert 1 / 0x01**

**Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangFrench**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Frankreich (Französisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangFrench** angegeben wurde.

## Kontakt

**\_LclSubLangFrenchBE**

**Ländereinstellungen Belgien (Französisch)**

**Wert 2 / 0x02**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangFrench**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Belgien (Französisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangFrench** angegeben wurde.

## Kontakt

**\_LclSubLangFrenchCA**

**Ländereinstellungen Kanada (Französisch)**

**Wert 3 / 0x03**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangFrench**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Kanada (Französisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangFrench** angegeben wurde.



## Kontakt

**\_LclSubLangFrenchCH** Ländereinstellungen  
Schweiz (Französisch) Wert 4 / 0x04

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangFrench

Option bei LocaleLoad() durch die die Ländereinstellungen der Schweiz (Französisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID LclLangFrench angegeben wurde.

## Kontakt

\_LclSubLangFrenchLU

Ländereinstellungen Luxemburg (Französisch)

Wert 5 / 0x05

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangFrench

Option bei LocaleLoad() durch die die Ländereinstellungen von Luxemburg (Französisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID LclLangFrench angegeben wurde.

## Kontakt

**\_LclSubLangFrenchMC**

**Ländereinstellungen Monaco (Französisch)**

**Wert 6 / 0x06**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangFrench**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Monaco (Französisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangFrench** angegeben wurde.

## Kontakt

**\_LclSubLangGerman**

**Ländereinstellungen Deutschland (Deutsch)**

**Wert 1 / 0x01**

**Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangGerman**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Deutschland (Deutsch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangGerman** angegeben wurde.

## Kontakt

\_LclSubLangGermanAT

Ländereinstellungen Österreich (Deutsch)

Wert 3 / 0x03

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangGerman

Option bei LocaleLoad() durch die die Ländereinstellungen von Österreich (Deutsch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID LclLangGerman angegeben wurde.

## Kontakt

\_LclSubLangGermanCH

Ländereinstellungen Schweiz (Deutsch)

Wert 2 / 0x02

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangGerman

Option bei LocaleLoad() durch die die Ländereinstellungen der Schweiz (Deutsch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID

LclLangGerman angegeben wurde.

## Kontakt

**\_LclSubLangGermanLI**

**Ländereinstellungen Liechtenstein (Deutsch)**

**Wert 5 / 0x05**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangGerman**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Liechtenstein (Deutsch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangGerman** angegeben wurde.

## Kontakt

\_LclSubLangGermanLU

Ländereinstellungen Luxemburg (Deutsch)

Wert 4 / 0x04

### Verwandte

Siehe Befehle,

LocaleLoad(),

LclLangGerman

Option bei LocaleLoad() durch die die Ländereinstellungen von Luxemburg (Deutsch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID LclLangGerman angegeben wurde.



**\_LclSubLangItalian**

**Ländereinstellungen Italien (Italienisch)**

**Wert 1 / 0x01**

**Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangItalian**

Option bei **LocaleLoad()** durch die die Ländereinstellungen von Italien (Italienisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangItalian** angegeben wurde.

## Kontakt

**\_LclSubLangItalianCH**

**Ländereinstellungen Schweiz (Italienisch)**

**Wert 2 / 0x02**

### **Verwandte**

Siehe **Befehle**,

**LocaleLoad()**,

**LclLangItalian**

Option bei **LocaleLoad()** durch die die Ländereinstellungen der Schweiz (Italienisch) geladen werden können. Diese Option kann nur angegeben werden, wenn als regionale ID **LclLangItalian** angegeben wurde.

## Kontakt

\_LclSubLangNeutral

Länderunabhängige Einstellungen

Wert 0 / 0x00

Verwandte

Siehe Befehle,

LocaleLoad()

Option bei LocaleLoad() durch die eine länderunabhängige Einstellung geladen werden kann. Spezielle Ländereinstellungen werden nicht geladen. Diese regionale Sub-ID kann nur in Verbindung mit \_LclLangNeutral angegeben werden.

## Kontakt

Befehle für zentrale Datenobjekte

Liste der Befehle und Konstanten zur Bearbeitung von zentrale Datenobjekten

Befehlsgruppen,  
Befehlsliste,

Siehe

Zentrale  
Datenobjekte

Befehle

- RmtDataRead
- RmtDataSearch
- RmtDataWrite

## Kontakt

**RmtDataRead(alpha1, int2,**  
**var alpha3) : int**



**Zentrales Datenobjekt lesen**

**alpha1** Schlüsselwert

**Optionen**

**0** Sperre nicht  
verändern

**int2** **RecLock** Datenobjekt sperren

**RecUnlock** Datenobjekt  
entsperren

Unter dem Schlüsselwert

**alpha3** abgelegter Wert

Fehlerwert

**rOk** Datenobjekt  
gelesen

**Resultat** **int** **rLocked** Datenobjekt  
gesperrt



**rNoRec** Datenobjekt nicht  
vorhanden

**Verwandte Befehle,**

**Siehe** **RmtDataWrite(),**  
**RmtDataSearch()**

Mit dieser Anweisung wird ein zentrales Datenobjekt ausgelesen. In (alpha1) wird der Schlüsselwert übergeben, der schon beim Schreiben des Objekts mit **RmtDataWrite()** übergeben wurde. Mit der Option (int2) kann angegeben werden, ob das Datenobjekt für andere Benutzer gesperrt werden soll. Eine Sperre verhindert das Überschreiben des Objekts. Bei der Übergabe von 0 wird die Sperre nicht verändert

In (alpha3) muss eine ausreichend lange alpha-Variable übergeben werden, um den gespeicherten Wert aufzunehmen. Ist die Variable zu kurz definiert, wird die Zeichenkette abgeschnitten.

Konnte das zentrale Datenobjekt gelesen werden, wird **rOk** zurückgegeben. Folgende Fehlerwerte können zurückgegeben werden:

**rLocked** Das zentrale Datenobjekt konnte gelesen aber nicht gesperrt werden.

**rNoRec** Das Datenobjekt zu dem angegebenen Schlüsselwert wurde nicht gefunden.

**Beispiel:**

```
local{ tRmtDataKey : alpha(250); tRmtDataValue : alpha(4096); tErg : int;}{ ... t
```

**Mögliche Laufzeitfehler:**

Die in (alpha1) übergebene Zeichenkette hat mehr als 2000

**ErrStringOverflow** Zeichen.

**RmtDataSearch(alpha1, int2) : int**        Schlüsselwerte der zentralen

Datenobjekte durchsuchen

**alpha1** Name des Schlüssels

**Leseoptionen**

**0** angegebenes  
Datenobjekt  
lesen

**RecFirst** erstes  
Datenobjekt  
lesen

**int2** **RecPrev** vorheriges  
Datenobjekt  
lesen

**RecNext** nächstes  
Datenobjekt  
lesen

**RecLast** letztes  
Datenobjekt  
lesen

**Resultat** **alpha** Schlüsselwert  
Verwandte Befehle,

Siehe

Zentrale Datenobjekte

Mit dieser Anweisung können zentrale Datenobjekte gesucht werden. Dazu wird ein Schlüsselbegriff in (alpha1) übergeben. Die Optionen entscheiden darüber, welcher Schlüsselbegriff zurückgegeben wird.

(int2) Aktion

**0** Der in (alpha1) angegebene Schlüsselwert wird gelesen. Ist dieser Wert nicht vorhanden, wird der nächst größere Wert zurückgegeben.

**RecFirst** Der erste Schlüsselwert wird zurückgegeben. Der Wert in (alpha1) wird ignoriert.

**RecPrev** Ausgehend vom in (alpha1) übergebenen Wert wird der vorhergehende Wert zurückgegeben.

**RecNext** Ausgehend vom in (alpha1) übergebenen Wert wird der nächste Wert zurückgegeben.

**RecLast** Der letzte Schlüsselwert wird zurückgegeben. Der Wert in (alpha1) wird ignoriert.

**Beispiel:**

```
// Alle Schlüsselwerte lesenfor aRmtDataKey # RmtDataSearch('', _RecFirst);loop aRmtDataKey #
```

**Mögliche Laufzeitfehler:**

**ErrStringOverflow** Der in (alpha1) angegebene Schlüsselwert ist länger als 2000 Zeichen.

## Kontakt

**RmtDataWrite(alpha1, int2,  
alpha3) : int**



**Zentrales Datenobjekt erzeugen**

**alpha1** Schlüsselwert

**Optionen**

**0** Sperre nicht ändern

**RecLock** Datenobjekt sperren

**int2** **RecUnlock** Datenobjekt

entsperren

**RmtDataTemp** temporäres

Datenobjekt

**alpha3** Daten

Fehlerwert

**rOk**

Datenobjekt

angelegt

**Resultat int** **rLocked** Datenobjekt 

gesperrt

**ErrLimitExceeded** Speicher

erschöpft

**Verwandte Befehle, RmtDataRead(),**

Siehe

**RmtDataSearch()**

Mit dieser Anweisung wird ein zentrales Datenobjekt angelegt oder überschrieben. Das zentrale Datenobjekt kann von jedem Client, der sich an der gleichen Datenbank angemeldet hat, abgerufen werden. Das Datenobjekt bleibt erhalten bis es mit einem Leerstring überschrieben wurde, sich der anlegende Benutzer von der Datenbank abgemeldet hat (bei Verwendung von **RmtDataTemp**) oder die Datenbank geschlossen wurde.

In (alpha1) wird ein Schlüsselbegriff angegeben. Der Schlüsselbegriff kann bis zu 2000 Zeichen lang und muss eindeutig sein. In den Optionen (int2) kann angegeben werden, ob das Datenobjekt nach dem Anlegen gleich gesperrt sein soll ( **RecLock**) oder nicht ( **RecUnlock**). Wird hier 0 Übergeben, wird die Sperre nicht verändert. Diese Option kann mit **RmtDataTemp** kombiniert werden, wenn das Objekt maximal bis zur Abmeldung des Benutzers erhalten bleiben soll.

Die Gesamtlänge von alpha1 und alpha3 (Schlüssel- und Datenanteil) darf 4601 Byte nicht überschreiten.

Konnte das zentrale Datenobjekt geschrieben werden, wird **rOk** zurückgegeben.

Folgende Fehlerwerte können zurückgegeben werden:

**rLocked** Das zentrale Datenobjekt ist von einem anderen Benutzer gesperrt.

**ErrLimitExceeded** Das Datenobjekt konnte nicht angelegt oder beschrieben werden, weil die maximale Speichergrenze (64 MB) erreicht ist.

**ErrStringOverflow** Der Schlüssel überschreitet eine Länge von 2000 Byte, die Daten eine Länge von 4096 Byte oder die Gesamtlänge überschreitet 4601 Byte.

## Kontakt

### Beispiele:

```
// Daten für einen Job-Server schreibenRmtDataWrite('JobNr1', _RecUnlock, 'Job Daten');// Eintrag
```



### Netzwerkinformationsbefehle

Befehle zum Ermitteln von Netzwerkinformationen Siehe

**Befehlsgruppen** **Befehlsliste**,

### Befehle

- NetInfo

### Konstanten

- NtiAddress
- NtiAddressServer
- NtiAddressTSC
- NtiConnectTime
- NtiName
- NtiNameIP
- NtiNameTSC
- NtiPing
- NtiPingTimeOut
- NtiProtocol
- NtiReadVol
- NtiRequests
- NtiWriteVol

## Kontakt



**NetInfo(int1[, alpha2]) : alpha**

**Netzwerkinformationen ermitteln**

**int1**      **Optionen (siehe Text)**

**Erweiterte Optionen (optional)**

**Rechnername oder IP-Adresse (bei**

**Verwendung von \_NtiPing oder**

**alpha2**    **\_NtiAddress)**

**Maximale Wartezeit (bei Verwendung**

**von \_NtiPingTimeOut)**

**Ermittlungsergebnis**

**Netzwerkinformation oder Resultat alpha**

**Fehlerwert bei Verwendung von**

**\_NtiPing oder \_NtiWakeOnLan**

Mit NetInfo() können diverse Netzwerkinformationen abgefragt werden. Dem Befehl wird die Informationsart, die in Erfahrung gebracht werden will, mitgegeben wie z. B. Rechnername, Netzwerkprotokoll, IP-Nummer des Rechners und Netzwerkadresse.

Folgende Optionen (int1) können verwendet werden:

- **\_NtiName** Rechnername

    ermitteln

- **\_NtiNameTSC**

    Rechnername (Terminal-Client) ermitteln

- **\_NtiNameIP** Host-Name

    ermitteln

- **\_NtiProtocol**

    Protokoll ermitteln

- **\_NtiAddress**

    Client-Netzwerkadresse (ohne alpha2) oder Adresse eines anderen Rechners (mit alpha2) ermitteln

- **\_NtiAddressTSC**

    Client-Netzwerkadresse (Terminal-Client) ermitteln

- **\_NtiAddressServer** Server-

    Netzwerkadresse ermitteln

- **\_NtiConnectTime**

    Verbindungszeit ermitteln

- **\_NtiRequests** Client-Anfrageanzahl

    ermitteln

- **\_NtiReadVol**

    Empfangenes Datenvolumen ermitteln

## Kontakt

- NtiWriteVol

Gesendetes Datenvolumen ermitteln

- NtiPing Rechner-Antwortzeit  
ermitteln

- NtiPingTimeOut Maximale  
Wartezeit setzen

- NtiWakeOnLan

Wake-on-LAN "Magic Packet" versenden

Resultate, die numerische Informationen enthalten müssen gegebenenfalls mit CnvIA() umgewandelt werden.

Konstanten für Netzwerkinformationsbefehle  
Konstanten für Netzwerkinformationsbefehle Siehe  
Netzwerkinformationsbefehle

- NtiAddress
- NtiAddressServer
- NtiAddressTSC
- NtiConnectTime
- NtiName
- NtiNameIP
- NtiNameTSC
- NtiPing
- NtiPingTimeOut
- NtiProtocol
- NtiReadVol
- NtiRequests
- NtiWriteVol

## Kontakt

\_NtiAddress

Client-Netzwerkadresse ermitteln

Wert 4

### Verwandte

Siehe Befehle,

### NetInfo()

Option bei NetInfo() durch die die Netzwerkadresse (IP-Adresse) des Clients ermittelt werden kann.

Wird bei NetInfo() der Parameter (alpha2) angegeben, wird versucht die IP-Adresse des angegebenen Rechners zu ermitteln. Kann keine IP-Adresse ermittelt werden, weil der Zielhost nicht erreicht werden kann, wird eine leere Zeichenkette zurückgegeben. Durch die Verwendung eines Präfixes vor dem Namen kann die Auswahl der IP-Adresse und somit des Protokolls gesteuert werden. Folgende Präfixe können angegeben werden:

ip4: ausschließlich IPv4

ip4f: bevorzugt IPv4, IPv6 wenn es keine IPv4-Adresse gibt

ip6: ausschließlich IPv6

ip6f: bevorzugt IPv6, IPv4 wenn es keine IPv6-Adresse gibt

Wird kein Präfix angegeben, wird automatisch ip4f: verwendet.

Das verwendete Protokoll kann mit der Option \_NtiProtocol ermittelt werden.

Beispiel:

```
// Arbeitsplatzabhängig Drucker setzenswitch (NetInfo(_NtiAddress)){ case '10.0.1.12' : aPrinter
```

Alternativ kann auch der Name des Client-Rechners mit der Option \_NtiName ermittelt werden.

## Kontakt

\_NtiAddressServer

Server-Netzwerkadresse ermitteln

Wert 5

Verwandte

Siehe Befehle,

NetInfo()

Option bei NetInfo() durch die die Netzwerkadresse (IP-Adresse) des Server ermittelt werden kann.

Das verwendete Protokoll kann mit der Option \_NtiProtocol ermittelt werden.

Beispiel:

```
// Server-abhängig temporären Pfad setzenswitch (NetInfo(_NtiAddressServer)){ case '10.0.0.1' :
```

## Kontakt

\_NtiAddressTSC

Client-Netzwerkadresse (Terminal-Client) ermitteln

Wert 13

### Verwandte

Siehe Befehle,

### NetInfo()

Option bei NetInfo() durch die die Netzwerkadresse des Terminal-Clients ermittelt werden kann, falls der CONZEPT 16-Client in einer Terminal-Sitzung läuft.

Bei Verwendung der Option \_NtiAddress wird nur die Adresse des Terminal-Servers ermittelt.

Zusätzlich kann auch der Name des Client-Rechners mit der Option \_NtiNameTSC ermittelt werden.

Der Befehl ist besonders in einer Terminal-Server-Umgebung nützlich, wenn die Applikation mit dem externen Debugger untersucht werden soll.

Der Debugger kann dazu auf der lokalen Maschine (nicht in der Terminal-Session) gestartet werden. Zum Verbinden mit der lokalen Maschine wird folgende Programmzeile ausgeführt:

```
DbgConnect (NetInfo (_NtiAddressTSC), false, false);
```

## Kontakt

**\_NtiConnectTime**

**Verbindungszeit ermitteln**

**Wert 6**

**Verwandte**

**Siehe Befehle,**

**NetInfo()**

**Option bei NetInfo() durch die die Verbindungszeit zwischen CONZEPT 16-Server und -Client in Sekunden ermittelt werden kann.**

**Um Rechenoperationen mit dem Rückgabewert durchführen zu können, muss dieser erst mit dem Befehl CnvIA() in einen ganzzahligen Wert gewandelt werden.**



## Kontakt

\_NtiName

Rechnername ermitteln

Wert 1

Verwandte

Siehe Befehle,

NetInfo()

Option bei NetInfo() durch die der Rechnername ermittelt werden kann. Unter Windows ist dies der NetBIOS-Name, bei Netware der Server-Name und bei Unix der Rechnername (uname).

Zusammen mit dem Namen kann mit der Option \_NtiNameIP noch der Host-Name ermittelt werden.

Beispiel:

```
// Arbeitsplatzabhängig Drucker setzenswitch (NetInfo(_NtiName)){ case 'OFFICE1' : aPrinterName
```

Alternativ kann auch die IP-Adresse des Client-Rechners mit der Option \_NtiAddress ermittelt werden.

## Kontakt

\_NtiNameIP

Host-Name ermitteln

Wert 2

Verwandte

Siehe Befehle,

NetInfo()

Option bei NetInfo() durch die der Host-Name des Rechners ermittelt werden kann. Der Host-Name wird entweder über die Hosts-Tabelle oder einen DNS ermittelt. Je nach Resolver ist die Domain-Information im Namen enthalten.

Beispiel:

```
NetInfo(_NtiNameIP); // -> 'support1.vectorsoft.de'
```

Soll nur der Name des Rechners oder seine IP-Adresse ermittelt werden, muss die Option

\_NtiName bzw. \_NtiAddress verwendet werden.

## Kontakt

\_NtiNameTSC

Rechnername (Terminal-Client) ermitteln

Wert 12

### Verwandte

Siehe Befehle,

### NetInfo()

Option bei NetInfo() durch die der Rechnername des Terminals ermittelt werden kann, falls der CONZEPT 16-Client in einer Terminal-Sitzung läuft.

Bei der Verwendung der Option \_NtiName würde in diesem Fall nur der Name des Terminal-Servers ermittelt werden. Läuft der CONZEPT 16-Client nicht in einer Terminal-Sitzung, wird mit \_NtiNameTSC und \_NtiName die gleiche Information ermittelt.

Zusätzlich kann auch die IP-Adresse des Client-Rechners mit der Option

\_NtiAddressTSC ermittelt werden.

## Kontakt

\_NtiPing

Rechner-Antwortzeit ermitteln

Wert 10

Verwandte

Siehe Befehle,

NetInfo()

Option bei NetInfo() durch die die Antwortzeit eines im Netzwerk befindlichen Rechners ermittelt werden kann.

Damit kann überprüft werden ob ein bestimmter Rechner im Netz erreichbar ist.

Das Resultat ist 0, wenn ein Timeout aufgetreten ist, bei positiven Werten entspricht dies der Antwortverzögerung in Millisekunden, negative Werte sind Fehlerwerte.

Standardmäßig wird nach drei Sekunden ein Timeout festgestellt. Die maximale Wartezeit kann mit der Option \_NtiPingTimeOut verändert werden.

Folgende Fehlerwerte können zurückgegeben werden:

<u>_ErrNetNoHost</u>	Host-Adresse konnte nicht ermittelt werden
<u>_ErrNetCreate</u>	ICMP-Sockets nicht verfügbar
<u>_ErrNetSelect</u>	Fehler bei Socket-Abfrage
<u>_ErrNetRead</u>	Fehler beim Lesen der Antwort
<u>_ErrNetReadLess</u>	Antwortpaket ungültig
<u>_ErrNetIcmpType</u>	Antwortpaket vom falschen Typ
<u>_ErrNetIcmpID</u>	Antwortpaket mit falscher ID
<u>_ErrNetWrite</u>	Anfrage konnte nicht versendet werden

## Kontakt

\_NtiPingTimeOut

Maximale Wartezeit setzen

Wert 11

Verwandte

Siehe Befehle,

NetInfo()

Option bei NetInfo() durch die die maximale Wartezeit auf eine Antwort eines Rechners in Millisekunden gesetzt werden kann.

Standardmäßig ist die maximale Wartezeit bei einer Anfrage mit der Option \_NtiPing auf 3 Sekunden eingestellt. Sie kann auf einen Wert zwischen 100 Millisekunden und 30 Sekunden verändert werden.

Beispiel:

```
// Server-Adresse ermitteltAdresse # NetInfo(_NtiAddressServer); // Maximale Wartezeit auf 100ms
```

## Kontakt

**\_NtiProtocol**

**Protokoll ermitteln**

**Wert 3**

**Verwandte**

**Siehe Befehle,**

**NetInfo()**

**Option bei NetInfo() durch die das zur Kommunikation mit dem Server verwendete Protokoll ermittelt werden kann.**

**Es wird nur die Kommunikation über TCP/IP unterstützt. Es wird immer 'TCP' zurückgegeben.**

## Kontakt

**\_NtiReadVol**

**Empfangenes Datenvolumen ermitteln**

**Wert 9**

### **Verwandte**

Siehe **Befehle**,

### **NetInfo()**

Option bei **NetInfo()** durch die die Größe des empfangenen Datenvolumens vom CONZEPT 16-Server seit dem Verbindungsaufbau in KB ermittelt werden kann.

Um Rechenoperationen mit dem Rückgabewert durchführen zu können, muss dieser erst mit dem Befehl **CnvIA()** in einen ganzzahligen Wert gewandelt werden.

## Kontakt

\_NtiRequests

Client-Anfrageanzahl ermitteln

Wert 7

Verwandte

Siehe Befehle,

NetInfo()

Option bei NetInfo() durch die die Anzahl der Anfragen des CONZEPT 16-Clients an den Server seit dem Verbindungsaufbau ermittelt werden kann.

Um Rechenoperationen mit dem Rückgabewert durchführen zu können, muss dieser erst mit dem Befehl CnvIA() in einen ganzzahligen Wert gewandelt werden.



## Kontakt

**\_NtiWriteVol**

**Gesendetes Datenvolumen ermitteln**

**Wert 8**

### **Verwandte**

Siehe **Befehle**,

### **NetInfo()**

Option bei **NetInfo()** durch die die Größe des gesendeten Datenvolumens an den CONZEPT 16-Server seit dem Verbindungsaufbau in KB ermittelt werden kann.

Um Rechenoperationen mit dem Rückgabewert durchführen zu können, muss dieser erst mit dem Befehl **CnvIA()** in einen ganzzahligen Wert gewandelt werden.

## Funktionen der Benutzeroberfläche

## Funktionen der Benutzeroberfläche

Siehe Befehlsgruppen,

### Befehlsliste

- Befehle für dynamische Objekte
- Deskriptor-Befehle
- Oberflächenobjektbefehle
- Objektinformationsbefehle

## Deskriptor-Befehle

Befehle zum Umgang mit Deskriptoren

Siehe Befehlsgruppen,

Befehlsliste

### Befehle

- HdlEnum
- HdlInfo
- HdlLink

### Konstanten

- HdlExists
- HdlSubType
- HdlType

obj -> HdEnum([int1]) :



handle

Deskriptor enumerieren

obj      Vorgänger-Deskriptor  
          Deskriptor-Typ

int1

(optional)

Resultat handle      Deskriptor  
          Verwandte Befehle,

Siehe

Deskriptoren

Mit dieser Funktion können alle aktiven Deskriptoren enumeriert werden.

Wird als Vorgänger-Deskriptor (obj) 0 angegeben, wird der erste Deskriptor, sonst der nachfolgende Deskriptor ermittelt. Aufgrund der internen Verwaltung der Deskriptoren werden diese nicht in numerisch aufsteigender Reihenfolge ermittelt.

Optional kann ein Deskriptor-Typ (int1) (siehe HdlType) angegeben werden, um nur Deskriptoren dieses Typs zu ermitteln.



Sollen alle Deskriptoren eines Typs entfernt werden, muss vor dem Entfernen der nächste Deskriptor ermittelt werden.

Resultate

Ist der Vorgänger-Deskriptor (obj) ungültig, oder kein weiterer Deskriptor vorhanden, wird 0 zurückgegeben. Sonst wird der nächste Deskriptor zurückgegeben.

Beispiele

```
// Alle Deskriptoren enumerierenfor tHdl # HdEnum(0);loop tHdl # tHdl->HdEnum();while (tHdl >
```

obj -> HdInfo(int1) : int  Deskriptor- 

Informationen ermitteln

obj      Deskriptor

Informationstyp

HdlExists      Deskriptor-Existenz

ermitteln

int1      HdlType      Deskriptor-Typ

ermitteln

HdlSubType Deskriptor-Untertyp

ermitteln

Resultat int Deskriptor-Information

Siehe Verwandte Befehle, Deskriptoren

Mit dieser Funktion können verschiedene Informationen über eine Deskriptor (obj) ermittelt werden. Die möglichen Rückgabewerte sind bei den Optionen beschrieben.

obj -> HdLink([int1]) : int  Deskriptor-

Verknüpfung ermitteln/erstellen

obj        Deskriptor

int1       Verknüpfung (optional)

Resultat int Deskriptor-Verknüpfung

Siehe Verwandte Befehle,

Deskriptoren, Blog

Mit dieser Funktion kann an den Deskriptor (obj) eine ganzzahlige Information (typischerweise ein weiterer Deskriptor) angehängen oder eine vorhandene Verknüpfung ermittelt werden. Zum Erstellen einer Verknüpfung muss (int1) gesetzt werden.

Mit diesem Befehl kann zum Beispiel eine einfach verkettete Liste von Oberflächen-Objekten aufgebaut werden, die zu einem späteren Zeitpunkt gemeinsam verarbeitet werden sollen. Zum Beispiel eine Liste von Objekten, deren Darstellung an bestimmte Ländereinstellungen angepasst werden müssen. Genauso kann an ein Oberflächen-Objekt der Deskriptor zu einem globalen Datenbereich, einer dynamischen Struktur oder einem Element aus einer dynamischen Struktur angebunden werden. So können zu jedem Objekt weitere Informationen zur Verfügung gestellt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor (obj) ungültig

Konstanten für Deskriptor-Befehle

Konstanten für Deskriptor-Befehle

Siehe Deskriptor-Befehle

- HdlExists
- HdlSubType
- HdlType

## Kontakt

**\_HdlExists**

**Deskriptor-Existenz ermitteln**

**Wert 0 / 0x00**

**Verwandte**

**Siehe Befehle,**

**HdlInfo()**

**Option bei HdlInfo() durch die die Existenz eines Deskriptors ermittelt werden kann.**

**Falls Deskriptor vorhanden: Resultat = 1, ansonsten Resultat = 0.**



### \_HdlSubType

Deskriptor-Untertyp ermitteln

Wert 2 / 0x02

#### Verwandte

Siehe Befehle,

#### HdlInfo()

Option bei HdlInfo() durch die der Untertyp eines Deskriptors ermittelt werden kann.

0 wird zurückgegeben, wenn es nicht um einen gültigen Deskriptor handelt, ansonsten einer der folgenden Werte:

- BinHdlDir (1)
- BinHdlObj (2)
- ChartDataHdlPie (2)
- ChartDataHdlPyramid (3)
- ChartDataHdlSurface (4)
- ChartDataHdlXy (1)
- ChartHdlPie (2)
- ChartHdlPyramid (3)
- ChartHdlSurface (4)
- ChartHdlXy (1)
- ComHdlArray (2)
- ComHdlObject (1)
- FsiHdlBufRead (2)
- FsiHdlBufWrite (4)
- FsiHdlDir (8)
- FsiHdlStd (1)
- HttpRecvRequest (2)
- HttpRecvResponse (4)
- HttpSendRequest (1)
- HttpSendResponse (3)
- JobProcess (2)
- JobThread (1)
- OdbcHdlApi (1)
- OdbcHdlClm (6)
- OdbcHdlCon (2)
- OdbcHdlStm (3)
- OdbcHdlStmParam (4)
- OdbcHdlTbl (5)
- StoHdlDir (1)
- StoHdlObj (2)
- TapiHdlDeviceItem (1)
- TapiHdlDeviceList (0)
- TapiHdlStatusItem (2)
- UrmTypeElmGroup (3)
- UrmTypeUser (1)
- UrmTypeUserGroup (2)

Wird der Untertyp von einem Datensatzpuffer ermittelt, wird die Dateinummer der dazugehörenden Datei zurückgegeben.

## Kontakt



Ist der Datensatz von einer untergeordneten Datei, die keine eigenen Feldpuffer hat, wird die Nummer der Hauptdatei ermittelt.

Beispiele:

```
switch (tHdl->HdlInfo(_HdlSubType)){ case _BinHdlDir : ... case _BinHdlObj : .....if (tHdl->Hd
```

### \_HdlType

Deskriptor-Typ ermitteln

Wert 1 / 0x01

#### Verwandte

Siehe Befehle,

#### HdlInfo()

Option bei HdlInfo() durch die der Typ eines Deskriptors ermittelt werden kann.

0 wird zurückgegeben, wenn es sich um keinen gültigen Deskriptor handelt, ansonsten einer der folgenden Werte:

- HdlBinary (17)
- HdlChart (39)
- HdlChartData (40)
- HdlCom (15)
- HdlCteItem (18)
- HdlCteList (19)
- HdlCteNode (34)
- HdlCteTree (20)
- HdlDataSpace (6)
- HdlDDE (7)
- HdlDDEItem (9)
- HdlDLL (13)
- HdlFile (1)
- HdlFilter (2)
- HdlFsiMonitor (24)
- HdlHTTP (31)
- HdlJob (36)
- HdlJobControl (37)
- HdlLocale (14)
- HdlMail (11)
- HdlMem (30)
- HdlMsx (26)
- HdlNewSelection (28)
- HdlOdbc (41)
- HdlPDF (33)
- HdlRecBuf (5)
- HdlSelection (4)
- HdlSocket (12)
- HdlSocketListen (22)
- HdlStorage (25)
- HdlSvcSocket (35)
- HdlSvcTime (27)
- HdlSystem (16)
- HdlTapi (23)
- HdlText (8)
- HdlTheme (45)
- HdlThemeSet (46)
- HdlTimer (10)
- HdlUrm (29)
- HdlWinObject (3)

- HdlXML (32)
- HdlXmlReader (43)
- HdlXmlWriter (44)

## Deskriptor-Typen

## Deskriptor-Typen

Siehe Deskriptor-Befehle,  
Deskriptoren

## Typen

- HdlBinary
- HdlChart
- HdlChartData
- HdlCom
- HdlCteItem
- HdlCteList
- HdlCteNode
- HdlCteTree
- HdlDataSpace
- HdlDDE
- HdlDDEItem
- HdlDLL
- HdlFile
- HdlFilter
- HdlFsiMonitor
- HdlHTTP
- HdlJob
- HdlJobControl
- HdlLocale
- HdlMail
- HdlMem
- HdlMsx
- HdlNewSelection
- HdlOdbc
- HdlPDF
- HdlRecBuf
- HdlSelection
- HdlSocket
- HdlSocketListen
- HdlStorage
- HdlSvcSocket
- HdlSvcTime
- HdlSystem
- HdlTapi
- HdlText
- HdlTheme
- HdlTimer
- HdlUrm
- HdlValidation
- HdlWinObject
- HdlXML
- HdlXmlReader
- HdlXmlWriter

## Untertypen

- BinHdlDir
- BinHdlObj
- ChartDataHdlPie
- ChartDataHdlPyramid
- ChartDataHdlSurface
- ChartDataHdlXy
- ChartHdlPie
- ChartHdlPyramid
- ChartHdlSurface
- ChartHdlXy
- ComHdlArray
- ComHdlObject
- FsiHdlBufRead
- FsiHdlBufWrite
- FsiHdlDir
- FsiHdlStd
- HttpRecvRequest
- HttpRecvResponse
- HttpSendRequest
- HttpSendResponse
- JobProcess
- JobThread
- OdbcHdlApi
- OdbcHdlClm
- OdbcHdlCon
- OdbcHdlStm
- OdbcHdlStmParam
- OdbcHdlTbl
- StoHdlDir
- StoHdlObj
- TapiHdlDeviceItem
- TapiHdlDeviceList
- TapiHdlStatusItem
- UrmTypeElmGroup
- UrmTypeUser
- UrmTypeUserGroup
- VldHdlDir
- VldHdlObj

**\_BinHdlDir**

**Verzeichnis-Deskriptor**

**Wert 1**

**Verwandte**

Siehe Befehle,

HdlInfo(),

BinDirOpen()

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Verzeichnisses hinweist.**

**\_BinHdlObj**

**Objekt-Deskriptor**

**Wert 2**

**Verwandte**

Siehe Befehle,

HdlInfo(),

BinOpen()

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines binären Objekts hinweist.**



**\_ComHdlArray**

**COM-Array-Deskriptor**

**Wert 2**

**Verwandte**

**Befehle,**

**Siehe HdlInfo(),**

**ComOpen(),**

**Array**

**Rückgabewert von HdlInfo() mit der Option \_HdlSubType der auf einen Deskriptor eines COM-Arrays hinweist.**

**\_ComHdlObject**

**COM-Objekt-Deskriptor**

**Wert 1**

**Verwandte**

Siehe Befehle,

HdlInfo(),

ComOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines COM-Objekts hinweist.

## Kontakt

**\_FsiHdlBufRead**

**Datei-Deskriptor**

**Wert 2**

### **Verwandte**

Siehe Befehle,

HdlInfo(),

FsiOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor einer Datei mit gepuffertem Lesezugriff hinweist.

Der Deskriptor muss mit dem Befehl FsiOpen() mit den Optionen FsiBuffer und FsiAcsR angelegt worden sein.

## Kontakt

**\_FsiHdlBufWrite**

**Datei-Deskriptor**

**Wert 4**

**Verwandte**

Siehe Befehle,

HdlInfo(),

FsiOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor einer Datei mit gepuffertem Schreibzugriff hinweist.

Der Deskriptor muss mit dem Befehl FsiOpen() mit den Optionen FsiBuffer und FsiAcsW angelegt worden sein.

**\_FsiHdlDir**

**Verzeichnis-Deskriptor**

**Wert 8**

**Verwandte**

Siehe Befehle,

HdlInfo(),

FsiDirOpen()

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Verzeichnisses hinweist.**

## Kontakt

\_FsiHdlStd

Datei-Deskriptor

Wert 1

Verwandte

Siehe Befehle,

HdlInfo(),

FsiOpen()

Rückgabewert von HdlInfo() mit der Option \_HdlSubType der auf einen Deskriptor einer Datei mit Standard-Optionen hinweist.

Der Deskriptor muss mit dem Befehl FsiOpen() mit der Option \_FsiStdRead oder \_FsiStdWrite angelegt worden sein.

**\_HdlBinary**

**Objekt/Verzeichnis-Deskriptor**

**Wert 17**

**Verwandte**

**Befehle,**

**Siehe HdlInfo(),**

**BinOpen(),**

**BinDirOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines binären Objekts/Verzeichnisses hinweist.**

**Über die Option \_HdlSubType kann ermittelt werden, ob es sich um ein binäres Objekt oder ein Verzeichnis handelt.**

**\_HdlChart**

**Chart-Objekt-Deskriptor**

**Wert 39**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**ChartOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Diagramm-Objekts hinweist.**



\_HdlChartData

ChartData-Objekt-Deskriptor

Wert 40

Verwandte

Siehe Befehle, HdlInfo(),

ChartDataOpen()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines ChartData-Objekts hinweist.

**\_HdlCom**

**COM-Objekt-Deskriptor**

**Wert 15**

**Verwandte**

Siehe Befehle,

HdlInfo(),

ComOpen()

Rückgabewert von HdlInfo() mit der Option HdlType der auf einen Deskriptor eines COM-Objekts hinweist.

**\_HdlCteItem**

**Element-Deskriptor**

**Wert 18**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**CteOpen()**

**Rückgabewert von HdlInfo() mit der Option HdlType der auf einen Deskriptor eines Elements einer dynamischen Struktur hinweist.**

**\_HdlCteList**

**Listenstruktur-Deskriptor**

**Wert 19**

**Verwandte**

Siehe Befehle,

HdlInfo(),

CteOpen()

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer Liste einer dynamischen Struktur hinweist.**

## Kontakt

**\_HdlCteNode**

**Knoten-Deskriptor**

**Wert 34**

**Verwandte**

Siehe Befehle,

HdlInfo(),

CteOpen()

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Knotens einer dynamischen Struktur hinweist.**

**\_HdlCteTree**

**Baumstruktur-Deskriptor**

**Wert 20**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**CteOpen()**

**Rückgabewert von HdlInfo() mit der Option HdlType der auf einen Deskriptor einer sortierten Liste einer dynamischen Struktur hinweist.**

**\_HdlDataSpace**

**Datenbereich-Deskriptor**

**Wert 6**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**VarAllocate()**

**Rückgabewert von HdlInfo() mit der Option HdlType der auf einen Deskriptor eines globalen Datenbereichs hinweist.**

**\_HdlDDE**

**DDE-Kanal-Deskriptor**

**Wert 7**

**Verwandte**

Siehe Befehle,

HdlInfo(),

DdeInit()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines DDE-Kanals hinweist.



## Kontakt

\_HdlDDEItem

DDE-Element-Deskriptor

Wert 9

Verwandte

Siehe Befehle,

HdlInfo()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines DDE-Elements hinweist.

**\_HdlDLL**

**DLL-Deskriptor**

**Wert 13**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**DllLoad()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer geladenen DLL hinweist.**

**\_HdlFile**

**Datei-Deskriptor**

**Wert 1**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**FsiOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer externen Datei hinweist.**

\_HdlFilter

Filter-Deskriptor

Wert 2

Verwandte

Siehe Befehle, HdlInfo(),

RecFilterCreate()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Datensatzfilters hinweist.

**\_HdlFsiMonitor**

**Deskriptor einer Dateiüberwachung**

**Wert 24**

**Verwandte**

**Siehe Befehle, HdlInfo(),**

**FsiMonitorOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer externen Dateiüberwachung hinweist.**

**\_HdlHTTP**

Deskriptor auf ein Http-Objekt

Wert 31

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**HttpOpen()**

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines HTTP-Objekts hinweist.

**\_HdlJob**

Deskriptor eines Job-Objekts

Wert 36

**Verwandte**

Siehe Befehle,

HdlInfo(),

Job

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Job-Objekts hinweist.

## Kontakt

**\_HdlJobControl**

Deskriptor eines JobControl-Objekts

Wert 37

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**JobControl**

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines JobControl-Objekt hinweist.



**\_HdlLocale**

**Locale-Objekt-Deskriptor**

**Wert 14**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**LocaleLoad()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Locale-Objekts hinweist.**

## Kontakt

**\_HdlMail**

**Mail-Objekt-Deskriptor**

**Wert 11**

**Verwandte**

Siehe **Befehle**,

**HdlInfo()**,

**MailOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Mail-Objekts hinweist.**

**\_HdlMem**

Deskriptor auf ein Memory-Objekt

Wert 30

**Verwandte**

Siehe Befehle,

HdlInfo(),

Memory

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Memory-Objekts hinweist.

## Kontakt

**\_HdlMsx**

**Message-Exchange-Deskriptor**

**Wert 26**

**Verwandte**

Siehe **Befehle**,

**HdlInfo()**,

**MsxOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Nachrichtenkanals hinweist.**

**\_HdlNewSelection**

Neuer Selektions-Deskriptor

Wert 28

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**SelCreate()**

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer dynamischen Selektion hinweist.

**\_HdlPDF**

**PDF-Objekt-Deskriptor**

**Wert 33**

**Verwandte**

Siehe Befehle,

HdlInfo(),

PdfOpen()

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines PDF-Objekts hinweist.**

**\_HdlOdbc**

**Deskriptor eines ODBC-Objekts**

**Wert 41**

**Verwandte Befehle,**

**Siehe HdlInfo(), Befehle für**

**ODBC-Verbindungen**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines OdbcApi-Objekts hinweist.**

## Kontakt

**\_HdlRecBuf**

Datensatzpuffer-Deskriptor

Wert 5

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**RecBufCreate()**

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Datensatzpuffers hinweist.

Die Dateinummer des Datensatzpuffers kann über den \_HdlSubType ermittelt werden.



**\_HdlSelection**

**Selektions-Deskriptor**

**Wert 4**

**Verwandte**

Siehe Befehle,

HdlInfo(),

SelOpen()

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer offenen Selektion hinweist.**

## Kontakt

**\_HdlSocket**

**Socket-Verbindungs-Deskriptor**

**Wert 12**

**Verwandte**

Siehe Befehle,

HdlInfo(),

SckConnect()

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor einer Socket-Verbindung hinweist.**

## Kontakt

**\_HdlSocketListen**

**Socket-Verbindungs-Deskriptor**

**Wert 22**

**Verwandte**

Siehe **Befehle**,

**HdlInfo()**,

**SckListen()**

**Rückgabewert von HdlInfo() mit der Option HdlType der auf einen Deskriptor einer passiven Socket-Verbindung hinweist.**

**\_HdlStorage**

Deskriptor eines Storage-Objekts


Wert 25

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**Storage-Objekte**

Rückgabewert von **HdlInfo()** mit der Option **\_HdlType** der auf einen Deskriptor eines Storage-Objekts  hinweist.

## Kontakt

\_HdlSvcSocket

Deskriptor eines Task mit Betriebsart SOCKET Wert  
35

### Verwandte

Siehe Befehle,

HdlInfo(),

mode

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Task mit Betriebsart SOCKET hinweist.

\_HdlSvcTime

Deskriptor eines Task mit Betriebsart TIME Wert

27

### Verwandte

Siehe Befehle,

HdlInfo(),

mode

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Task mit Betriebsart TIME hinweist.

**\_HdlSystem**

**System-Objekt-Deskriptor**

**Wert 16**

**Verwandte**

Siehe **Befehle**,

**HdlInfo()**,

**\_Sys**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines System-Objekts hinweist.**

## Kontakt

**\_HdlTapi**

**Tapi-Objekt-Deskriptor**

**Wert 23**

**Verwandte**

Siehe Befehle,

HdlInfo(),

TapiOpen()

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Tapi-Objekts hinweist.**



**\_HdlText**

**Text-Deskriptor**

**Wert 8**

**Verwandte**

Siehe **Befehle**,

**HdlInfo()**,

**TextOpen()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines geöffneten Texts hinweist.**

## Kontakt

\_HdlTheme

Theme-Deskriptor

Wert 45

Verwandte

Siehe Befehle, HdlInfo(),

WinThemeOpen()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Theme-Objekts hinweist.

**\_HdlTimer**

**Timer-Deskriptor**

**Wert 10**

**Verwandte**

**Siehe Befehle, HdlInfo(),**

**SysTimerCreate()**

**Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Timer-Ereignisses hinweist.**

## Kontakt

\_HdlUrm

Deskriptor auf ein Objekt der Benutzerverwaltung Wert 29

### Verwandte

Siehe Befehle,

HdlInfo(),

UrmOpen()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Objekts der Benutzerverwaltung hinweist. Genauere Informationen werden bei der Ermittlung des \_HdlSubType. Dort wird dann eine der Konstanten \_UrmTypeUser, \_UrmTypeUserGroup oder \_UrmTypeElmGroup zurückgegeben.

## Kontakt

\_HdlValidation

Deskriptor eines Validierungsverzeichnisses bzw. Validierungselementes Wert 47

Verwandte

Befehle,

Siehe HdlInfo(),

VldOpen(),

VldDirOpen()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines Validierungsverzeichnisses bzw. Validierungselementes hinweist.

Über die Option \_HdlSubType kann ermittelt werden, ob es sich um ein Validierungselement oder ein Verzeichnis handelt.

\_HdlWinObject

Oberflächen-Objekt-Deskriptor

Wert 3

Verwandte

Siehe Befehle,

HdlInfo(),

App

Rückgabewert von HdlInfo() mit der Option HdlType der auf einen Deskriptor eines Oberflächen-Objekts oder des App-Objekts hinweist.

\_HdlXML

XML-Objekt-Deskriptor

Wert 32

Verwandte

Siehe Befehle,

HdlInfo(),

XmlLoad()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines XML-Objekts hinweist.

## Kontakt

\_HdlXmlReader

XmlReader-Deskriptor

Wert 43

Verwandte

Siehe Befehle, HdlInfo(),

XmlOpenReader()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines XmlReader-Objekts hinweist.



## Kontakt

\_HdlXmlWriter

XmlWriter-Deskriptor

Wert 44

Verwandte

Siehe Befehle, HdlInfo(),

XmlOpenWriter()

Rückgabewert von HdlInfo() mit der Option \_HdlType der auf einen Deskriptor eines XmlWriter-Objekts hinweist.

**\_OdbcHdlApi**

**ODBC-Schnittstellen-Deskriptor**

**Wert 1**

**Verwandte**

Siehe Befehle,

HdlInfo(),

OdbcOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines ODBC-Schnittstellen-Objekts hinweist.

**\_OdbcHdlCln**

**ODBC-Spalten-Deskriptor**

**Wert 6**

**Verwandte Befehle,**

**Siehe HdlInfo(),**

**OdbcCatalogCln()**

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines ODBC-Spalten-Objekts hinweist.**

**\_OdbcHdlCon**

**ODBC-Verbindungs-Deskriptor**

**Wert 2**

**Verwandte Befehle,**

Siehe **HdlInfo()**,

**OdbcConnect()**,

**OdbcConnectDriver()**

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines ODBC-Verbindungs-Objekts hinweist.**

## Kontakt

**\_OdbcHdlStm**

**ODBC-Statement-Deskriptor**

**Wert 3**

**Verwandte Befehle,**

**Siehe HdlInfo(),**

**OdbcExecuteDirect()**

**Rückgabewert von HdlInfo() mit der Option HdlSubType, der auf einen Deskriptor eines ODBC-Statement-Objekts hinweist, dass mit OdbcExecuteDirect() angelegt wurde.**

**\_OdbcHdlStmParam**

**ODBC-Statement-Deskriptor**

**Wert 4**

**Verwandte**

Siehe Befehle,

HdlInfo(),

OdbcPrepare()

Rückgabewert von HdlInfo() mit der Option HdlSubType, der auf einen Deskriptor eines ODBC-Statement-Objektes hinweist, dass mit OdbcPrepare() angelegt wurde.

## Kontakt

**\_OdbcHdlTbl**

**ODBC-Tabellen-Deskriptor**

**Wert 5**

**Verwandte**

**Siehe Befehle, HdlInfo(),**

**OdbcCatalogTbl()**

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines ODBC-Tabellen-Objekts hinweist.**

**\_StoHdlDir**

**Storage-Verzeichnis**

**Wert 1**

**Verwandte**

Siehe Befehle,

HdlInfo(),

StoDirOpen()

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Storage-Verzeichnisses hinweist.**



**\_StoHdlObj**

Storage-Objekt

Wert 2

**Verwandte**

Siehe Befehle,

HdlInfo(),

StoOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Storage-Objekts hinweist.

## Kontakt

**\_TapiHdlDeviceItem**

Deskriptor auf ein Cte-Item mit TAPI-Geräteinformationen Wert 1

### **Verwandte**

Siehe Befehle,

HdlInfo(),

TapiOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Cte-Items mit Geräteinformationen über ein TAPI-Gerät hinweist. Die Liste wurde mit der Anweisung TapiOpen() gefüllt.

## Kontakt

**\_TapiHdlDeviceList**

Deskriptor auf eine Cte-Liste mit TAPI-Geräteinformationen Wert 0

### **Verwandte**

Siehe Befehle,

HdlInfo(),

TapiOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Cte-Liste mit Geräteinformationen über TAPI-Geräte hinweist. Die Liste wurde mit der Anweisung TapiOpen() erstellt und gefüllt.

## Kontakt

**\_TapiHdlStatusItem**

Deskriptor auf ein Cte-Item mit TAPI-Statusinformationen Wert 2

### **Verwandte**

Siehe Befehle,

HdlInfo(),

TapiInfo()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Cte-Items mit Statusinformationen über ein TAPI-Gerät hinweist. Die Liste wurde mit der Anweisung TapiInfo() gefüllt.

**\_VldHdlDir**

Deskriptor eines Validierungsverzeichnisses Wert 1

### **Verwandte**

Siehe Befehle,

HdlInfo(),

VldDirOpen()

Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Validierungsverzeichnisses hinweist.

**\_VldHdlObj**

**Deskriptor eines Validierungselementes**

**Wert 2**

**Verwandte**

Siehe Befehle,

**HdlInfo()**,

**VldOpen()**

**Rückgabewert von HdlInfo() mit der Option HdlSubType der auf einen Deskriptor eines Validierungselementes hinweist.**

### Objektinformationsbefehle

Befehle zum Ermitteln von Objektinformationen Siehe

**Befehlsgruppen** **Befehlsliste**,








### Befehle

- WinInfo

### Konstanten

- WinApplication
- WinClientHeight
- WinClientWidth
- WinContextMenu
- WinCount
- WinDynamic
- WinErrorCode
- WinFirst
- WinFocusKey
- WinFrame
- WinFrameActive
- WinFrameForeground
- WinHighContrast
- WinIndex
- WinInteriorHeight
- WinInteriorWidth
- WinItem
- WinLast
- WinLstEditObject
- WinMenu
- WinNext
- WinNodeMouseSelect
- WinObject
- WinObjectActive
- WinObjectMaximized
- WinParent
- WinPrev
- WinRoot
- WinScreenBPP
- WinScreenDpiX
- WinScreenDpiY
- WinScreenHeight
- WinScreenWidth
- WinState
- WinThemeActive
- WinType
- WinUnicode

## Kontakt

obj -> WinInfo(int1[, int2, int3]) : int        Oberflächen-  
Objekt-Informationen ermitteln  
obj      Oberflächenobjekt  
int1      Optionen (siehe Text)  
int2      Zähler (optional)  
int3      Objekttyp (optional)  
Resultat int Objektinformation (siehe Text)  
          WinSearch(), HdlInfo(),  
Siehe      OrderPass, Blog

Mit diesem Befehl können verschiedene Informationen zu dem übergebenen Objekt (obj) oder zu der Umgebung, in der die Applikation gestartet wurde, ermittelt werden. Zudem können relativ zum angegebenen Objekt, andere Objekte ermittelt werden. Die unterschiedlichen Funktionen werden durch die Übergabe entsprechender Konstanten in (int1) aufgerufen.

Die Funktionen können in folgende Bereiche aufgeteilt werden:

- Navigation durch den Objektbaum
- Objekte nach Typ iterieren
- Ermitteln anderer Objekte
- Informationen über das Objekt
- Informationen über die Umgebung

Navigation durch den Objektbaum

Werden die folgenden Konstanten übergeben, ist die Rückgabe des Befehls immer der Deskriptor eines Objekts. In (obj) kann ein beliebiges Oberflächenobjekt übergeben werden.

- WinFirst

Das Resultat ist das erste untergeordnete Objekt des Objekts (obj).

- WinLast

Das Resultat ist das letzte untergeordnete Objekt des Objekts (obj).

- WinNext

Das Resultat ist das nachfolgende Objekt des Objekts (obj).

- WinPrev

Das Resultat ist das vorhergehende Objekt des Objekts (obj).

- WinParent

Das Resultat ist das übergeordnete Objekt des Objekts (obj).

- WinRoot

Das Resultat ist das Wurzel-Objekt des Objekts (obj).

- WinFrame

Das Resultat ist das übergeordnete Frame-Objekt des Objekts (obj).



## Kontakt



Die Reihenfolge, in der über die Parameter WinFirst, WinPrev, WinNext und WinLast zugegriffen wird, entspricht der Erstellungsreihenfolge der einzelnen Objekte. Die Intervall-Objekte innerhalb des GanttGraph-Objekts werden vom Betriebssystem nicht in einer Liste verwaltet. Die Reihenfolge beim Lesen der Intervalle kann somit von der Erstellungsreihenfolge abweichen. Bei den Objekten Notebook, RecList und DataList kann alternativ über die Eigenschaft OrderPass die Anzeigereihenfolge ermittelt werden.

In (int3) kann ein Objekt-Typ angegeben werden. Als Werte stehen die bei WinType beschriebenen Konstanten zur Verfügung. Der Objekt-Typ wird nur in Kombination mit den Optionen WinFirst, WinPrev, WinNext, WinLast und WinParent ausgewertet. Es wird das entsprechende Objekt des angegebenen Typs zurückgegeben.

Beispiel:

```
// Übergeordnetes Notebook-Objekt ermitteltObj->WinInfo(_WinParent, 0, _WinTypeNotebook);
```

Objekte nach Typ iterieren

Werden Objekte nach Typ iteriert, stellt der übergebene Objekttyp dabei nicht immer den exakten Objekttyp dar, sondern eine Objektgruppe.

Beispiel:

```
for tChild # tParent->WinInfo(_WinFirst, 1, _WinTypeEdit);loop tChild # tChild->WinInfo(_WinNext
```

Die Schleife ermittelt nicht nur Edit-Objekte vom Typ WinTypeEdit, sondern auch DateEdit, FloatEdit, TextEdit, etc, da diese zur Objektgruppe der Eingabeobjekte gehören.

Die folgende Schleife ermittelt dagegen nur DateEdit-Objekte, da diese keine eigene Objektgruppe bilden.

```
for tChild # tParent->WinInfo(_WinFirst, 1, _WinTypeDateEdit);loop tChild # tChild->WinInfo(_Win
```

Folgende Objektgruppen existieren:

Eingabeobjekte ( <u>_WinTypeEdit</u> )	<u>Edit</u> , <u>IntEdit</u> , <u>BigIntEdit</u> , <u>FloatEdit</u> , <u>DecimalEdit</u> , <u>TimeEdit</u> , <u>DateEdit</u> , <u>ColorEdit</u> , <u>FontNameEdit</u> , <u>FontSizeEdit</u> , <u>TextEdit</u>
Frame-Objekte ( <u>_WinTypeFrame</u> )	<u>Frame</u> , <u>AppFrame</u> , <u>MdiFrame</u> , <u>TrayFrame</u> , <u>Dialog</u>
Schaltflächen ( <u>_WinTypeButton</u> )	<u>Button</u> , <u>ColorButton</u> , <u>Checkbox</u> , <u>Radiobutton</u>
Schalter ( <u>_WinTypeCheckBox</u> )	<u>Checkbox</u> , <u>Radiobutton</u>
Label-Objekte ( <u>_WinTypeLabel</u> )	<u>Label</u> , <u>HyperLink</u>
Toolbar-Objekte ( <u>_WinTypeToolbar</u> )	<u>Toolbar</u> , <u>Statusbar</u>
PrintForm-Objekte ( <u>_PrtTypePrintForm</u> )	<u>PrintForm</u> , <u>PrintDoc</u> , <u>PrintDocRecord</u> , <u>PrintFormList</u>

Sollen nur Objekte des angegebenen Typs ermittelt werden, muss die Konstante WinTypeExact mit dem Objekttyp verodert werden:

## Kontakt

```
for tChild # tParent->WinInfo(_WinFirst, 1, _WinTypeEdit | _WinTypeExact);loop tChild # tChild-
```

Die Konstante wird bei den Optionen WinFirst, WinLast, WinNext, WinPrev und WinParent ausgewertet und bei allen anderen Optionen ignoriert.

### Ermitteln anderer Objekte

Bei diesen Funktionen werden bestimmte Objekte ermittelt. Als Übergabeparameter müssen entweder bestimmte Objekttypen oder 0 übergeben werden. Welche Objekte übergeben werden können, ist bei den Konstanten erläutert.

- WinApplication

Das Resultat ist das Application-Objekt. In (obj) kann 0 oder ein beliebiges Objekt angegeben werden.

- WinFrameActive

Das Resultat ist das aktive MDIFrame-Objekt des in (obj) übergebenen AppFrame-Objekts.

- WinObject

Das Resultat ist ein bestimmtes Unterobjekt des Objekts (obj). Das übergebene Objekt muss von einem bestimmten Typ sein.

- WinObjectActive

Das Resultat ist das aktive GroupTile-Objekt des in (obj) übergebenen GroupSplit-Objekts.

- WinObjectMaximized

Das Resultat ist das maximierte GroupTile-Objekt des in (obj) GroupSplit-Objekts.

- WinMenu

Das Resultat ist das Menü des in (obj) übergebenen Frame bzw. AppFrame-Objekts.

- WinContextMenu

Das Resultat ist das Kontextmenü-Objekt des Objekts (obj).

- WinLstEditObject

Wird in einem RecList- oder DataList-Objekt ein Eintrag editiert, wird mit dieser Konstante das Editier-Objekt zurückgegeben.

- WinFrameForeground

Das Resultat ist das Vordergrundfenster. Als Objekt (obj) muss 0 als Zähler (int2) muss 0 oder 1 angegeben werden.

### Informationen über das Objekt

In dieser Gruppe von Konstanten werden Informationen über das in (obj) angegebene Objekt zurückgegeben. Bei einigen Konstanten können nur bestimmte Objekttypen

## Kontakt

übergeben werden. Dies ist bei den Konstanten erläutert.

- WinCount

Das Resultat ist die Anzahl der untergeordneten Objekte des Objekts (obj).

- WinClientHeight

Es wird die Höhe des Clientbereichs (inklusive Tool- und Windowbars) zurückgegeben. Es können nur Objekte vom Typ Frame, AppFrame oder MdiFrame übergeben werden.

- WinClientWidth

Es wird die Breite des Clientbereichs (inklusive Tool- und Windowbars) zurückgegeben. Es können nur Objekte vom Typ Frame, AppFrame oder MdiFrame übergeben werden.

- WinInteriorHeight

Es wird die Höhe des Anzeigebereiches ohne Menü, Tool- und Windowbars zurückgegeben. Bei AppFrame-Objekten entspricht das der Höhe des Bereiches, in dem MdiFrame-Objekte dargestellt werden können. Es können nur Objekte vom Typ Frame bzw. AppFrame übergeben werden.

- WinInteriorWidth

Es wird die Breite des Anzeigebereiches ohne Menü, Tool- und Windowbars zurückgegeben. Bei AppFrame-Objekten entspricht das der Breite des Bereiches, in dem MdiFrame-Objekte dargestellt werden können. Es können nur Objekte vom Typ Frame bzw. AppFrame übergeben werden.

- WinType

Das Resultat ist der Typ des Objekts (obj).

- WinFocusKey

Das Resultat ist die Taste, mit der das Objekt verlassen wurde.

- WinIndex

Das Resultat ist die Position des Objekts (obj) innerhalb des übergeordneten Objekts.

- WinItem

Das Resultat ist die Position der Datenzeile des Spalten-Objekts (obj) eines DataList-Objekts.

- WinState

Das Resultat ist der Status des Frame/AppFrame/MDIFrame- oder des GroupTile-Objekts (obj).

- WinDynamic

Wird ein Objekt (obj) angegeben, gibt der Befehl zurück, ob es dynamisch erstellt wurde (1) oder nicht (0). Ein Objekttyp (int3) darf in diesem Fall nicht angegeben werden.

Wird als Objekt 0 und in (int3) ein Objekttyp angegeben, gibt der Befehl zurück,

## Kontakt

ob der Objekttyp mit WinCreate() erzeugt werden kann.

- WinUnicode

Wird als Resultat 1 zurückgegeben, unterstützt das Objekt (obj) Unicode-Eigenschaften, bei 0 nicht. Die betreffenden Eigenschaften sind auf der Seite Unicode-Unterstützung ersichtlich.

## Informationen über die Umgebung

Mit diesen Konstanten werden Parameter der Umgebung abgefragt. Außer bei der Ermittlung des Fehlerwertes muss kein Objekt übergeben werden. Anstelle des Deskriptors wird der Wert 0 angegeben.

`WinInfo(0, int1[, int2, int3])`

- WinScreenWidth

Das Resultat ist die horizontale Bildschirmauflösung.

- WinScreenHeight

Das Resultat ist die vertikale Bildschirmauflösung.

- WinScreenDpiX

Das Resultat ist die horizontale Bildschirmauflösung in DPI.

- WinScreenDpiY

Das Resultat ist die vertikale Bildschirmauflösung in DPI.

- WinScreenBBP

Das Resultat ist die aktuelle Bildschirmfarbtiefe.

- WinThemeActive

Das Resultat ist die Aktivität der betriebssystemabhängigen Darstellung.

- WinHighContrast

Das Resultat ist 1, wenn Windows im Modus "Hoher Kontrast" läuft, sonst 0.

- WinErrorCode

Das Resultat ist der Windows-Fehlerwert. Der Fehlerwert muss nur nach der Installation zur Darstellung von Office-Dateien (InstallCtxOffice) abgefragt werden.

- WinNodeMouseSelect

Das Resultat ist die Maustaste, mit der zuletzt ein TreeNode in einem TreeView-Objekt selektiert wurde.

## Beispiel 1:

```
// Alle Objekte eines Fensters ermitteln@A+@C+sub AllObj( aStartObj : handle;) local { tObj
```

## Beispiel 2:

## Kontakt

```
// Alle Einträge eines Menüs ermitteln// Menü des Fensters ermittelntMenu # tFrame->WinInfo(_WinM
```

### Beispiel 3:

```
// Bestimmte Objekte eines Gantt-Graphs ermittelnsInfo( aGanttGraph : handle; // Übergebene
```

Der Parameter (int2) wird nur in Kombination mit den Optionen WinPrev, WinNext und WinParent ausgewertet. Mit dem Zähler kann eine Schrittweite übergeben werden.

### Beispiel:

```
// Übernächstes Objekt ermittelntObj->WinInfo(_WinNext, 2); // Großvater-Objekt ermittelntObj->Win
```

Bei der Angabe von negative Werten in (int2) liefert der Befehl den Wert 0 zurück.

### Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Oberflächenobjekt (obj) ungültig oder muss 0 sein. Bei der Option <u>WinDynamic</u> wurde sowohl ein Objekt (obj) als auch ein Objekttyp (int3) angegeben.
<u>ErrValueInvalid</u>	Bei der Option <u>WinFrameForeground</u> wurde ein Zähler (int2) < 0 oder > 1 angegeben.

Konstanten für Objektinformationsbefehle

Konstanten für Objektinformationsbefehle

Siehe Objektinformationsbefehle

- WinApplication
- WinClientHeight
- WinClientWidth
- WinContextMenu
- WinCount
- WinDynamic
- WinErrorCode
- WinFirst
- WinFocusKey
- WinFrame
- WinFrameActive
- WinFrameForeground
- WinHighContrast
- WinIndex
- WinInteriorHeight
- WinInteriorWidth
- WinItem
- WinLast
- WinLstEditObject
- WinMenu
- WinNext
- WinNodeMouseSelect
- WinObject
- WinObjectActive
- WinObjectMaximized
- WinParent
- WinPrev
- WinRoot
- WinScreenBPP
- WinScreenDpiX
- WinScreenDpiY
- WinScreenHeight
- WinScreenWidth
- WinState
- WinThemeActive
- WinType
- WinUnicode

## Kontakt

**\_WinApplication**

Application-Objekt ermitteln

Wert 3000

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die das Application-Objekt ermittelt werden kann.

**\_WinClientHeight**

Höhe des Clientsbereichs

Wert 1005

Verwandte Befehle,

Siehe WinInfo(),

WinClientWidth,

WinInteriorHeight

Option bei WinInfo() durch die die Höhe des Clientbereichs eines Fensters ermittelt werden kann. Der Clientbereich ist die Größe des Fensters ohne Titel- und Menüzeile, sowie Fensterrahmen. Eventuell vorhandene Tool-, Window- oder Statusbars sind im Clientbereich enthalten.



**\_WinClientWidth**

**Breite des Client-Bereichs**

**Wert 1004**

**Verwandte Befehle,**

Siehe WinInfo(),

**WinClientHeight,**

**WinInteriorWidth**

**Option bei WinInfo() durch die die Breite des Clientbereichs eines Fensters ermittelt werden kann.**

**Der Clientbereich ist die Größe des Fensters ohne Titel- und Menüzeile, sowie Fensterrahmen.**

**Eventuell vorhandene Tool-, Window- oder Statusbars sind im Clientbereich enthalten.**

## Kontakt

**\_WinContextMenu**

**Kontextmenü-Objekt ermitteln**

**Wert 12**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das Kontextmenü-Objekt eines Objekts ermittelt werden kann.

Mit dieser Konstante wird innerhalb des Ereignisses **EvtMenuInitPopup** der Deskriptor des aktuellen Kontextmenüs (sofern in dem Argument aMenuItem 0 übergeben wurde) ermittelt.

## Kontakt

**\_WinCount**

Anzahl untergeordneter Objekte ermitteln

Wert 9

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die die Anzahl der unmittelbar untergeordneten Objekte ermittelt werden kann.

Beim GanttGraph-Objekt ist dies die Anzahl der Intervalle, beim AppFrame die Anzahl der MdiFrames, beim RecList-, RecView- und DataList-Objekt die Anzahl der Spalten. Beim Spalten-Objekt vom RecView wird die Anzahl der SubColumns ermittelt.

## Kontakt

**\_WinDynamic**

Ermitteln, ob das Objekt dynamisch erstellt wurde Wert 18

### Verwandte

Siehe Befehle,

WinInfo(),

WinCreate()

Option bei WinInfo() durch die ermittelt werden kann, ob das Objekt dynamisch erstellt wurde.

Das Ergebnis ist 1, wenn das Objekt dynamisch erstellt wurde und 0, wenn nicht.

**\_WinErrorCode**

**Windows-Fehlerwert**

**Wert 2500**

**Verwandte**

Siehe Befehle,

WinInfo(),

InstallCtxOffice

Option bei WinInfo() durch der Windows-Fehlerwert ermittelt werden kann. Es muss kein Objekt übergeben werden.

## Kontakt

**\_WinFirst**

Erstes untergeordnetes Objekt ermitteln

Wert 4

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das erste untergeordnete Objekt eines Objekts ermittelt werden kann.

Die Reihenfolge wird bei den Objekten **NoteBook**, **DataList** und **RecList** durch die Eigenschaft **OrderPass** beeinflusst.

## Kontakt

**\_WinFocusKey**

Taste ermitteln, mit dem das Objekt verlassen wurde Wert  
2003

### Verwandte

Siehe Befehle,

### WinInfo()

Option bei WinInfo() durch die die Taste ermittelt werden kann, die dazu führte, dass das Objekt den Fokus verloren hat.

**Beispiel:**

```
switch (aEvt:Obj->WinInfo(_WinFocusKey)){ case _WinKeyTab | _WinKeyShift : { // Shift + TAB
```

**\_WinFrame**

**Fenster-Objekt ermitteln**

**Wert 10**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Option bei WinInfo() durch die das Fenster-Objekt eines Objektes ermittelt werden kann. Dabei wird ausgehend vom übergebenen Objekt solange das Eltern-Objekt ermittelt, bis ein Objekt gefunden wird, das vom Typ AppFrame, MdiFrame oder Frame ist.**

**Wird der Deskriptor eines Fenster-Objektes übergeben, wird dieser auch zurückgegeben.**



## Kontakt

**\_WinFrameActive**

Aktives MDI-Fenster-Objekt ermitteln

Wert 2001

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die das aktive MDI-Fenster-Objekt eines Applikations-Fenster-Objekts ermittelt werden kann.

## Kontakt

\_WinFrameForeground

Vordergrundfenster ermitteln

Wert 2005

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die das Vordergrundfenster ermittelt werden kann. Es handelt sich dabei um das Fenster, mit dem der Anwender gerade arbeitet.

Die Funktion liefert immer den Deskriptor eines Frame-Objektes zurück und unterscheidet sich somit vom Befehl WinFocusGet(), der das Objekt mit dem Eingabefokus liefert.

Im Falle eines AppFrame-Objektes mit untergeordneten MdiFrame-Objekten, liefert die Funktion das AppFrame-Objekt. Zur Ermittlung des aktiven MdiFrame kann WinInfo() mit der Option WinFrameActive aufgerufen werden.

Die Funktion liefert 0 zurück, wenn kein Vordergrundfenster vorhanden ist oder es sich nicht um ein von CONZEPT 16 erstelltes Fenster handelt.

Beispiele:

```
// Vordergrundfenster ermitteltWin # WinInfo(0, _WinFrameForeground); // Vordergrundfenster nur z
```

## Kontakt

**\_WinHighContrast**

Prüfen, ob Windows im Modus "Hoher Kontrast" läuft Wert  
1010

### Verwandte

Siehe Befehle,

### WinInfo()

Option bei WinInfo() durch die ermittelt werden kann, ob Windows im Modus "Hoher Kontrast" läuft.



Sofern eine Anwendung unter Modern Theme Style läuft, wird empfohlen, das Theme '\_WindowsColor', oder ein Theme, welches auf diesem basiert, zu verwenden.

### Beispiel

```
// Aktuelles Theme abhängig vom Windows-Modus "Hoher Kontrast" aktivierenif (WinInfo(0, _WinHighC
```

## Kontakt

**\_WinIndex**

**Objektposition ermitteln**

**Wert 11**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Option bei WinInfo() durch die die Position eines Objekts innerhalb des übergeordneten Objekts ermittelt werden kann.**

## Kontakt

**\_WinInteriorHeight**

Höhe des Bereiches, in dem MDI-Fenster dargestellt werden können Wert 1009

### **Verwandte**

Siehe Befehle, WinInfo(),

**WinInteriorWidth**

**WinClientHeight**

Option bei WinInfo(), durch die die Höhe des Anzeigebereiches ohne Menü, Tool-, Status- und Windowbars ermittelt werden kann. Bei AppFrame-Objekten entspricht das der Höhe des Bereiches, in dem MdiFrame-Objekte dargestellt werden können.

**Beispiel:**

```
tInteriorHeight # $AppFrm->WinInfo(_WinInteriorHeight);
```

## Kontakt

**\_WinInteriorWidth**

Breite des Bereiches, in dem MDI-Fenster dargestellt werden können Wert 1008

**Verwandte Befehle,**

Siehe WinInfo(),

**WinInteriorHeight**

**WinClientWidth**

Option bei WinInfo(), durch die die Breite des Anzeigebereiches ohne Menü, Tool-, Status- und Windowbars ermittelt werden kann. Bei AppFrame-Objekten entspricht das der Breite des Bereiches, in dem MdiFrame-Objekte dargestellt werden können.

**Beispiel:**

```
tInteriorWidth # $AppFrm->WinInfo(_WinInteriorWidth);
```

## Kontakt

\_WinItem

Datenzellenposition ermitteln

Wert 14

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die die Position der Datenzelle eines Spalten-Objektes eines DataList-Objektes ermittelt werden kann.

Die Position der Datenzelle kann bei den Befehlen WinLstCellSet() und WinLstCellGet() verwendet werden.

## Kontakt

**\_WinLast**

**Letztes untergeordnetes Objekt ermitteln**

**Wert 5**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Option bei WinInfo() durch die das letzte untergeordnete Objekt eines Objekts ermittelt werden kann.**

**Die Reihenfolge wird bei den Objekten Notebook, DataList und RecList durch die Eigenschaft OrderPass beeinflusst.**



## Kontakt

**\_WinLstEditObject**

**Editier-Objekt ermitteln**

**Wert 2004**

### Verwandte

Siehe Befehle,

WinInfo(),

WinLstEdit()

Option bei WinInfo() durch die das Editier-Objekt bei einer editierten RecList- oder DataList-Objekt ermittelt werden kann.

## Kontakt

**\_WinMenu**

**Menü-Objekt ermitteln**

**Wert 8**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das Menü-Objekt eines **Fenster-** / **Applikations-Fenster-**oder **MenuButton-**Objekts ermittelt werden kann.

## Kontakt

**\_WinNext**

Nachfolgendes Objekt ermitteln

Wert 7

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das nachfolgende Objekt eines Objekts ermittelt werden kann.

Die Reihenfolge wird bei den Objekten **Notebook**, **DataList** und **RecList** durch die Eigenschaft **OrderPass** beeinflusst.

## Kontakt

\_WinNodeMouseSelect

Maustaste ermitteln, mit der der Knoten selektiert wurde Wert 2006

### Verwandte

Siehe Befehle,

### WinInfo()

Option bei WinInfo() durch die ermittelt werden kann, ob und mit welcher Taste ein TreeNode in einem TreeView-Objekt ausgewählt wurde. Der Wert wird jeweils vor dem Auslösen des Ereignisses EvtNodeSelect gesetzt. Bei aktivierter Mehrfachauswahl kann sich die Information auch auf das Selektieren mehrerer Knoten bzw. deselektieren von Knoten beziehen.

Der ermittelte Wert kann mit den folgenden Werten und Konstanten verglichen werden:

- \_WinMouseLeft - Selektionsaktion mit der linken Maustaste
- \_WinMouseRight - Selektionsaktion mit der rechten Maustaste
- 0 - Selektionsaktion unbekannt (Tastatur oder prozedural z. B. über CurrentInt-Eigenschaft).



Der Wert 0 wird auch zurückgegeben, wenn das TreeView erstmalig den Fokus erhält, kein TreeNode selektiert ist und dann mit der rechten Maustaste ein TreeNode selektiert wird.

\_WinObject

Bestimmtes Objekt ermitteln

Wert 15

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die ein bestimmtes Objekt ermittelt werden kann.

Bei einem GanttGraph kann das View-Objekt ermittelt werden.

Bei einem GroupSplit-Objekt kann über die Optionen \_WinObjectActive und \_WinObjectMaximized das Aktive bzw. Maximierte GroupTile-Objekt ermittelt werden.

Beispiele:

```
tView # tGanttGraph->WinInfo(_WinObject, <View-Nummer>, _WinTypeGanttView);
```

Ist bei einem GanttGraph-Objekt die Eigenschaft SplitStyle nicht auf \_WinSplitNone gesetzt, können durch ziehen des "Splitters" (über bzw. links vom Scrollbalken) ein, zwei oder vier Ansichten (Views) erzeugt werden.

Die Nummern der Ansichten werden von links nach rechts und von oben nach unten vergeben. Ist also ein GanttGraph-Objekt horizontal in zwei Ansichten geteilt, hat die obere die Nummer 1 und die untere die Nummer 2.

```
// Ermitteln, ob ein GroupTile-Objekt maximiert ist. tObj # gGroupSplit->WinInfo(_WinObject, _Win
```

## Kontakt

**\_WinObjectActive**

Aktives GroupTile-Objekt ermitteln

Wert 1

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die das aktive GroupTile-Objekt eines GroupSplit-Objekts ermittelt werden kann.

**Beispiel**

```
// Aktives GroupTile-Objekt ermittelntObj # gGroupSplit->WinInfo(_WinObject, _WinObjectActive);
```

## Kontakt

**\_WinObjectMaximized**

Maximiertes GroupTile-Objekt ermitteln

Wert 2

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das maximierte **GroupTile**-Objekt eines **GroupSplit**-Objekts ermittelt werden kann.

```
// Ermitteln, ob ein GroupTile-Objekt maximiert isttObj # gGroupSplit->WinInfo(_WinObject, _WinOb
```

## Kontakt

**\_WinParent**

Übergeordnetes Objekt ermitteln

Wert 3

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das übergeordnete Objekt eines Objekts ermittelt werden kann.



## Kontakt

\_WinPrev

Vorhergehendes Objekt ermitteln

Wert 6

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die das vorhergehende Objekt eines Objekts ermittelt werden kann.

Die Reihenfolge wird bei den Objekten Notebook, DataList und RecList durch die Eigenschaft OrderPass beeinflusst.

**\_WinRoot**

Wurzel-Objekt ermitteln

Wert 2

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die das Wurzel-Objekt eines Objekts ermittelt werden kann. Ausgehend vom übergebenen Objekt wird solange das Eltern-Objekt ermittelt, bis das Objekt erreicht wird, das kein Eltern-Objekt hat. Das Objekt wird zurückgegeben. Hierbei handelt es sich in der Regel um ein **AppFrame-** oder **Frame-**Objekt. Wird das Wurzel-Objekt eines **MdiFrame** abgefragt, wird im **EvtInit** der **MdiFrame** zurückgegeben.

**\_WinScreenBPP**

**Bildschirmfarbtiefe ermitteln**

**Wert 1002**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Option bei WinInfo() durch die die aktuelle Bildschirmfarbtiefe in Bit ermittelt werden kann.**

**8 Bit = 256 Farben**

**15 Bit = HiColor**

**16 Bit = HiColor**

**24 Bit = TrueColor**

**32 Bit = TrueColor**

**\_WinScreenDpiX**

**Horizontale DPI-Auflösung des Bildschirms**

**Wert 1006**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Option bei WinInfo() durch die die horizontale Bildschirmauflösung in DPI (Dots per inch) ermittelt werden kann.**

**\_WinScreenDpiY**

**Vertikale DPI-Auflösung des Bildschirms**

**Wert 1007**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Option bei WinInfo() durch die die vertikale Bildschirmauflösung in DPI (Dots per inch) ermittelt werden kann.**

**\_WinScreenHeight**

Vertikale Bildschirmauflösung ermitteln

Wert 1001

**Verwandte**

Siehe Befehle,

**WinInfo()**,

**AreaScreen**

Option bei WinInfo() durch die die vertikale Bildschirmauflösung des Primärbildschirms in Pixel ermittelt werden kann.

**\_WinScreenWidth**

Horizontale Bildschirmauflösung ermitteln

Wert 1000

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**AreaScreen**

Option bei **WinInfo()** durch die die horizontale Bildschirmauflösung des Primärbildschirms in Pixel ermittelt werden kann.

## Kontakt

\_WinState

Fensterstatus ermitteln

Wert 2000

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die der Status eines

Fensters/Applikations-Fensters/MDI-Fensters oder eines GroupTile-Objektes ermittelt werden kann.

Folgende Fenster-Statuswerte können zurückgegeben werden:

\_WinDialogMaximized Fenster maximiert

\_WinDialogMinimized Fenster minimiert

\_WinDialogNormal Fenster weder maximiert noch minimiert

Beispiel:

```
if (tFrame->WinInfo(_WinState) = _WinDialogMinimized){ // Dialog minimiert}
```

Folgende GroupTile-Statuswerte können zurückgegeben werden:

\_WinStateAttachMaximized GroupTile maximiert

\_WinStateAttachClosed GroupTile geschlossen

\_WinStateAttachNormal GroupTile weder maximiert noch geschlossen

Beispiel:

```
if (tGroupTile->WinInfo(_WinState) = _WinStateAttachClosed){ // GroupTile geschlossen}
```



## Kontakt

**\_WinThemeActive**

Aktivität der betriebssystemabhängigen Darstellung ermitteln Wert 1003

### Verwandte

Siehe Befehle,

### WinInfo()

Option bei WinInfo() durch die die Aktivität der betriebssystemabhängigen Darstellung ermittelt werden kann.

Bei Verwendung der betriebssystemabhängigen Darstellung ist das Resultat größer 0.

**Beispiel:**

```
// Betriebssystem-Darstellung aktivif (WinInfo(0, _WinThemeActive) > 0){ ...}// Betriebssystem-
```

\_WinType

Objekt-Typ ermitteln

Wert 1

Verwandte

Siehe Befehle,

WinInfo()

Option bei WinInfo() durch die der Typ eines Objekts ermittelt werden kann.

Folgende Objekt-Typen können zurückgegeben werden:

- WinTypeAnimation
- WinTypeAppFrame
- WinTypeApplication
- WinTypeBarcode
- WinTypeBigIntEdit
- WinTypeButton
- WinTypeC16Info
- WinTypeCalendar
- WinTypeCanvas
- WinTypeCanvasGraphic
- WinTypeCheckBox
- WinTypeChromium
- WinTypeCodeEdit
- WinTypeCodeEditList
- WinTypeCodeEditListColumn
- WinTypeColorButton
- WinTypeColorEdit
- WinTypeComArguments
- WinTypeComFileOpen
- WinTypeComFileSave
- WinTypeComFont
- WinTypeComOdbc
- WinTypeComPath
- WinTypeComPrint
- WinTypeComPrintSetup
- WinTypeCtxAdobeReader
- WinTypeCtxDocEdit
- WinTypeCtxDocEditRuler
- WinTypeCtxDocEditTBar
- WinTypeCtxOffice
- WinTypeDataList
- WinTypeDataListPopup
- WinTypeDateEdit
- WinTypeDecimalEdit
- WinTypeDialog
- WinTypeDivider
- WinTypeDocView
- WinTypeDragDataFormat
- WinTypeDragDataObject
- WinTypeEdit
- WinTypeFloatEdit

- WinTypeFontNameEdit
- WinTypeFontSizeEdit
- WinTypeFrame
- WinTypeFrameClient
- WinTypeGanttAxis
- WinTypeGanttGraph
- WinTypeGanttView
- WinTypeGroup
- WinTypeGroupBox
- WinTypeGroupColumn
- WinTypeGroupSplit
- WinTypeGroupTile
- WinTypeGroupTileButton
- WinTypeHelpTip
- WinTypeHyperLink
- WinTypeIcon
- WinTypeIntEdit
- WinTypeInterval
- WinTypeIvlBox
- WinTypeIvlLine
- WinTypeLabel
- WinTypeListColumn
- WinTypeMdiFrame
- WinTypeMenu
- WinTypeMenuButton
- WinTypeMenuItem
- WinTypeMetaPicture
- WinTypeNotebook
- WinTypeNotebookPage
- WinTypePicture
- WinTypePopupColor
- WinTypePopupList
- WinTypePpcObject
- WinTypePrintDevice
- WinTypePrinter
- WinTypePrinterList
- WinTypePrintJob
- WinTypeProgress
- WinTypePrtJobPreview
- WinTypePrtPpvControl
- WinTypePrtPreviewDlg
- WinTypeRadioButton
- WinTypeRecList
- WinTypeRecListPopup
- WinTypeRecNavigator
- WinTypeRecView
- WinTypeRTFEdit
- WinTypeScrollbar
- WinTypeScrollbox
- WinTypeSelDataObject
- WinTypeStatusbar

- WinTypeStatusBarButton
- WinTypeStoList
- WinTypeStoListPopup
- WinTypeTextEdit
- WinTypeTimeEdit
- WinTypeToolbar
- WinTypeToolbarButton
- WinTypeToolbarDock
- WinTypeToolbarMenu
- WinTypeToolbarRTF
- WinTypeTrayFrame
- WinTypeTreeNode
- WinTypeTreeView
- WinTypeWebNavigator
- WinTypeWindowbar

**\_WinUnicode**

Unicode-Unterstützung des Objektes ermitteln

Wert 19

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Option bei **WinInfo()** durch die die **Unicode-Unterstützung** des Objektes ermittelt werden kann. Bei einem Resultat von 1 besitzt das Objekt Unicode-Eigenschaften. Welche Eigenschaften dies betrifft, kann der Seite **Unicode-Unterstützung** entnommen werden.

## Objekttypen

### Objekttypen

Siehe Objektinformationsbefehle

- WinTypeAnimation
- WinTypeAppFrame
- WinTypeApplication
- WinTypeBarcode
- WinTypeBigIntEdit
- WinTypeButton
- WinTypeC16Info
- WinTypeCalendar
- WinTypeCanvas
- WinTypeCanvasGraphic
- WinTypeCheckBox
- WinTypeChromium
- WinTypeCodeEdit
- WinTypeCodeEditList
- WinTypeCodeEditListColumn
- WinTypeColorButton
- WinTypeColorEdit
- WinTypeComArguments
- WinTypeComFileOpen
- WinTypeComFileSave
- WinTypeComFont
- WinTypeComOdbc
- WinTypeComPath
- WinTypeComPrint
- WinTypeComPrintSetup
- WinTypeCtxAdobeReader
- WinTypeCtxDocEdit
- WinTypeCtxDocEditRuler
- WinTypeCtxDocEditTBar
- WinTypeCtxOffice
- WinTypeDataList
- WinTypeDataListPopup
- WinTypeDateEdit
- WinTypeDecimalEdit
- WinTypeDialog
- WinTypeDivider
- WinTypeDocView
- WinTypeDragDataFormat
- WinTypeDragDataObject
- WinTypeEdit
- WinTypeFloatEdit
- WinTypeFontNameEdit
- WinTypeFontSizeEdit
- WinTypeFrame
- WinTypeFrameClient
- WinTypeGanttAxis
- WinTypeGanttGraph

- WinTypeGanttView
- WinTypeGroup
- WinTypeGroupBox
- WinTypeGroupColumn
- WinTypeGroupSplit
- WinTypeGroupTile
- WinTypeGroupTileButton
- WinTypeHelpTip
- WinTypeHyperLink
- WinTypeIcon
- WinTypeIntEdit
- WinTypeInterval
- WinTypeIvlBox
- WinTypeIvlLine
- WinTypeLabel
- WinTypeListColumn
- WinTypeMdiFrame
- WinTypeMenu
- WinTypeMenuButton
- WinTypeMenuItem
- WinTypeMetaPicture
- WinTypeNotebook
- WinTypeNotebookPage
- WinTypePicture
- WinTypePopupColor
- WinTypePopupList
- WinTypePpcObject
- WinTypePrintDevice
- WinTypePrinter
- WinTypePrinterList
- WinTypePrintJob
- WinTypeProgress
- WinTypePrtJobPreview
- WinTypePrtPpvControl
- WinTypePrtPreviewDlg
- WinTypeRadioButton
- WinTypeRecList
- WinTypeRecListPopup
- WinTypeRecNavigator
- WinTypeRecView
- WinTypeRTFEdit
- WinTypeScrollbar
- WinTypeScrollbox
- WinTypeSelDataObject
- WinTypeStatusbar
- WinTypeStatusbarButton
- WinTypeStoList
- WinTypeStoListPopup
- WinTypeTextEdit
- WinTypeTimeEdit
- WinTypeToolbar

- WinTypeToolBarButton
- WinTypeToolBarDock
- WinTypeToolBarMenu
- WinTypeToolBarRTF
- WinTypeTrayFrame
- WinTypeTreeNode
- WinTypeTreeView
- WinTypeWebNavigator
- WinTypeWindowbar



## Kontakt

**\_WinTypeAnimation**

**Animations-Objekt**

**Wert 61.865.986 /  
0x03B00002**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Animation**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Animation-Objekt hinweist.**

**\_WinTypeAppFrame**

**Applikations-Fenster-Objekt**

Wert 51.380.225 /  
0x03100001

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**AppFrame**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **AppFrame**-Objekt hinweist.

## Kontakt

**\_WinTypeApplication**

**Applikations-Objekt**

Wert <sup>2/</sup>

**0x00000002**

**Verwandte**

Siehe Befehle,

WinInfo(),

Application

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Application-Objekt hinweist.

## Kontakt

**\_WinTypeBigIntEdit**

**BigInt-Eingabe-Objekt**

Wert 54.525.964 /  
0x0340000C

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**BigIntEdit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **BigIntEdit**-Objekt hinweist.

## Kontakt

**\_WinTypeButton**

**Schaltflächen-Objekt**

Wert 53.477.377 /  
0x03300001

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Button**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Button**-Objekt hinweist.

**\_WinTypeC16Info**

**CONZEPT 16-Dialog "Info"**

Wert 65.011.760 /  
0x03E00030

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**WinOpen( WinC16Info,**

**...)**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf den Info-Dialog von CONZEPT 16 hinweist.

## Kontakt

**\_WinTypeCalendar**

**Kalender-Objekt**

Wert 61.865.987 /  
0x03B00003

**Verwandte**

Siehe Befehle,

WinInfo(),

Calendar

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Calendar-Objekt hinweist.

## Kontakt

\_WinTypeCanvas

Canvas-Objekt

Wert 56.623.105 /  
0x03600001

Verwandte

Siehe Befehle,

WinInfo(),

Canvas

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Canvas-Objekt hinweist.



\_WinTypeCanvasGraphic

CanvasGraphic-Objekt

Wert 56.688.640 /  
0x03610000

Verwandte

Siehe Befehle,

WinInfo(),

CanvasGraphic

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein CanvasGraphic-Objekt hinweist.

**\_WinTypeCheckBox**

**Mehrfachauswahlknopf-Objekt**

Wert 53.477.378 /  
0x03300002

**Verwandte**

Siehe Befehle,

WinInfo(),

Checkbox

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Checkbox-Objekt hinweist.

## Kontakt

\_WinTypeCodeEdit

CodeEdit-Objekt

Wert 54.528.013 / 0x0340080D

Verwandte Befehle, WinInfo(),

Siehe \_WinTypeCodeEditList,

\_WinTypeCodeEditListColumn,

CodeEdit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein CodeEdit-Objekt hinweist.

**\_WinTypeCodeEditList**

Autovervollständigungsliste des CodeEdit-Objektes

Wert 54.528.016 / 0x03400810

**Verwandte Befehle,**

Siehe \_WinTypeCodeEdit,

\_WinTypeCodeEditListColumn,

CodeEdit

Auswahl der Autovervollständigungsliste eines CodeEdit-Objektes zum Setzen von Theme-Eigenschaften.

## Kontakt

**\_WinTypeCodeEditListColumn**

Spalte der Autovervollständigungsliste des CodeEdit-Objektes

Wert 54.528.017 /

0x03400811

**Verwandte Befehle,**

Siehe WinTypeCodeEdit,

WinTypeCodeEditList,

CodeEdit

Auswahl einer Spalte der Autovervollständigungsliste eines CodeEdit-Objektes zum Setzen von Theme-Eigenschaften.

**\_WinTypeColorButton**

**Farbschaltflächen-Objekt**

Wert 53.477.384 /  
0x03300008

**Verwandte**

Siehe Befehle,

WinInfo(),

ColorButton

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein ColorButton-Objekt hinweist.

## Kontakt

**\_WinTypeColorEdit**

**Farbeingabe-Objekt**

Wert 54.525.959 /  
0x03400007

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**ColorEdit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **ColorEdit**-Objekt hinweist.

## Kontakt

**\_WinTypeComArguments**

**Argument-Objekt der COM-Schnittstelle**

Wert 134.217.729 /

**0x08000001**

**Verwandte**

Siehe Befehle,

WinInfo(),

EvtCtxEvent

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Argument-Objekt der COM-Schnittstelle hinweist. Diese Objekt wird als Argument aComArguments beim Ereignis EvtCtxEvent übergeben.**



## Kontakt

**\_WinTypeComFileOpen**

**Common-Dialog (\_WinComFileOpen)**

Wert 65.011.728 /

**0x03E00010**

**Verwandte Befehle,**

Siehe **WinInfo()**,

**WinComFileOpen**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein Common-Dialog vom Typ **\_WinComFileOpen** hinweist.

## Kontakt

**\_WinTypeComFileSave**

**Common-Dialog (\_WinComFileSave)**

Wert 65.011.729 /

**0x03E00011**

**Verwandte**

Siehe Befehle, WinInfo(),

WinComFileSave

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Common-Dialog vom Typ \_WinComFileSave hinweist.

**\_WinTypeComFont**

**Common-Dialog (\_WinComFont)**

Wert 65.011.733 /  
0x03E00015

**Verwandte**

Siehe Befehle,

WinInfo(),

WinComFont

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Common-Dialog vom Typ \_WinComFont hinweist.

## Kontakt

**\_WinTypeComOdbc**

**Common-Dialog (\_WinComOdbc)**

Wert 65.011.734 /  
0x03E00016

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**WinComOdbc**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein Common-Dialog vom Typ **\_WinComOdbc** hinweist.

**\_WinTypeComPath**

**Common-Dialog (\_WinComPath)**

Wert 65.011.730 /  
0x03E00012

**Verwandte**

Siehe Befehle,

WinInfo(),

WinComPath

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Common-Dialog vom Typ \_WinComPath hinweist.

**\_WinTypeComPrint**

**Common-Dialog (\_WinComPrint)**

Wert 65.011.731 /

0x03E00013

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**WinComPrint**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein Common-Dialog vom Typ **\_WinComPrint** hinweist.

**\_WinTypeComPrintSetup**

**Common-Dialog (\_WinComPrintSetup)**

Wert 65.011.732 /

**0x03E00014**

**Verwandte Befehle,**

Siehe **WinInfo()**,

**WinComPrintSetup**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein Common-Dialog vom Typ **\_WinComPrintSetup** hinweist.

## Kontakt

**\_WinTypeCtxAdobeReader**

**Kontroll-Objekt des Adobe Readers**

Wert 59.768.850 /  
0x03900012

**Verwandte**

Siehe Befehle,

WinInfo(),

CtxAdobeReader

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein CtxAdobeReader-Objekt hinweist.



## Kontakt

**\_WinTypeCtxDocEdit**

**CtxDocEdit-Objekt**

Wert 59.768.855 /  
0x03900017

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**CtxDocEdit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **CtxDocEdit**-Objekt hinweist.

**\_WinTypeCtxDocEditRuler**

**Lineal eines CtxDocEdit-Objektes**

Wert 59.768.854 /

0x03900016

**Verwandte Befehle,**

**WinInfo(),**

**Siehe CtxDocEdit,**

**WinToolbarRulerH,**

**WinToolbarRulerV**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Lineal eines CtxDocEdit-Objektes hinweist.**

**\_WinTypeCtxDocEditTBar**

**Toolbar eines CtxDocEdit-Objektes**

**Wert 59.768.857 /**

**0x03900019**

**Verwandte Befehle,**

**Siehe WinInfo(),**

**CtxDocEdit,**

**WinToolbarButtons**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf eine Toolbar eines CtxDocEdit-Objektes hinweist.**

## Kontakt

**\_WinTypeCtxOffice**

**Kontroll-Objekt für Microsoft Office Dokumente**

Wert 59.768.849 /

0x03900011

**Verwandte**

Siehe Befehle,

WinInfo(),

CtxOffice

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein CtxOffice-Objekt hinweist.

## Kontakt

**\_WinTypeDataList**

**DataList-Objekt**

Wert 57.671.937 /  
0x03700101

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**DataList**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **DataList**-Objekt hinweist.

**\_WinTypeDataListPopup**

**DataListPopup-Objekt**

Wert 57.672.193 /  
0x03700201

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**DataListPopup**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **DataListPopup**-Objekt hinweist.

## Kontakt

**\_WinTypeDateEdit**

**Datumseingabe-Objekt**

Wert 54.525.957 /  
0x03400005

**Verwandte**

Siehe Befehle,

WinInfo(),

DateEdit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein DateEdit-Objekt hinweist.

**\_WinTypeDecimalEdit**

**Decimal-Eingabe-Objekt**

Wert 54.525.965 /  
0x0340000D

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**DecimalEdit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **DecimalEdit**-Objekt hinweist.



## Kontakt

**\_WinTypeDialog**

**Dialog-Objekt**

Wert 50.331.652 /  
0x03000004

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Frame**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Frame**-Objekt hinweist.

## Kontakt

**\_WinTypeDivider**

**Trennlinien-Objekt**

Wert 52.428.806 /  
0x03200006

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Divider**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Divider**-Objekt hinweist.

## Kontakt

**\_WinTypeDocView**

**Dokument-Objekt**

Wert 61.866.004 /  
0x03B00014

**Verwandte**

Siehe Befehle,

WinInfo(),

DocView

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein DocView-Objekt hinweist.

**\_WinTypeDragDataFormat**

**Datenformat-Objekt**

Wert 100.663.298 /

**0x06000002**

**Verwandte**

Siehe Befehle,

WinInfo(),

DragDataFormat

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein DragDataFormat-Objekt hinweist.

**\_WinTypeDragDataObject**

**Daten einer Drag & Drop-Operation**

**Wert** 100.663.297

/ 0x06000001

**Verwandte**

Siehe Befehle,

WinInfo(),

DragData

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein DragData-Objekt hinweist.**

**\_WinTypeEdit**

**Alpha-Eingabe-Objekt**

Wert 54.525.953 /  
0x03400001

**Verwandte**

Siehe Befehle,

WinInfo(),

Edit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Edit-Objekt hinweist.

**\_WinTypeFloatEdit**

**Float-Eingabe-Objekt**

Wert 54.525.955 /  
0x03400003

**Verwandte**

Siehe Befehle,

WinInfo(),

FloatEdit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein FloatEdit-Objekt hinweist.

**\_WinTypeFontNameEdit**

**Schriftnameneingabe-Objekt**

Wert 62.914.576 /  
0x03C00010

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**FontNameEdit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **FontNameEdit**-Objekt hinweist.



**\_WinTypeFontSizeEdit**

**Schriftgrößeneingabe-Objekt**

**Wert 62.914.577 /**

**0x03C00011**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**FontSizeEdit**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein FontSizeEdit-Objekt hinweist.**

**\_WinTypeFrame**

**Frame-Objekt**

Wert 50.331.651 /  
0x03000003

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Frame**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Frame**-Objekt hinweist.



Da für die Anzeige zur Laufzeit alle Frames in den Dialogtyp konvertiert werden, ist der Rückgabewert von **WinInfo()** auf Fenster-Objekte immer **\_WinTypeDialog**.

## Kontakt

**\_WinTypeFrameClient**

**FrameClient-Objekt**

Wert 60.817.424 /  
0x03A00010

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**FrameClient**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **FrameClient**-Objekt hinweist.

**\_WinTypeGanttAxis**

**Achsen-Objekt**

Wert 16.777.220 /  
**0x01000004**

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**Axis,**

**GanttGraph**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Axis**-Objekt eines **GanttGraph**-Objektes hinweist.

## Kontakt

**\_WinTypeGanttGraph**

**Gantt-Diagramm-Objekt**

Wert 61.866.000 /  
0x03B00010

**Verwandte**

Siehe Befehle,

WinInfo(),

GanttGraph

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein GanttGraph-Objekt hinweist.

**\_WinTypeGanttView**

**View-Objekt**

Wert 61.866.001 /  
0x03B00011

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**View,**

**GanttGraph**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **View**-Objekt eines **GanttGraph**-Objektes hinweist.

**\_WinTypeGroup**

**RecView-Gruppe**

Wert 61.866.012 /  
**0x03B0001C**

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**Group**

**RecView**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Group**-Objekt eines **RecView**-Objektes hinweist.

## Kontakt

**\_WinTypeGroupBox**

**Gruppen-Objekt**

Wert 52.428.803 /

**0x03200003**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**GroupBox**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **GroupBox**-Objekt hinweist.



**\_WinTypeGroupColumn**

**Spalten-Objekt**

**Wert 61.866.008 /  
0x03B00018**

**Verwandte**

**Befehle,**

**Siehe WinInfo(),**

**Column**

**RecView**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Column-Objekt eines RecView-Objektes hinweist.**

## Kontakt

**\_WinTypeGroupSplit**

**GroupSplit-Objekt**

Wert 52.428.807 /  
0x03200007

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**GroupSplit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **GroupSplit**-Objekt hinweist.

## Kontakt

**\_WinTypeGroupTile**

**GroupTile-Objekt**

Wert 52.428.808 /  
**0x03200008**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**GroupTile**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **GroupTile**-Objekt hinweist.

**\_WinTypeGroupTileButton**

**GroupTile-Button-Objekt**

Wert 60.817.676 /  
**0x03A0010C**

**Verwandte**

Siehe Befehle, WinInfo(),

GroupTile-Button

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein GroupTile-Button-Objekt hinweist.

## Kontakt

**\_WinTypeHyperLink**

**HyperLink-Objekt**

Wert 52.428.802 /  
0x03200002

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**HyperLink**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **HyperLink**-Objekt hinweist.

**\_WinTypeIcon**

**Symbol-Objekt**

**Wert 61.865.989 /  
0x03B00005**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Icon**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Icon-Objekt hinweist.**

**\_WinTypeIntEdit**

**Int-Eingabe-Objekt**

Wert 54.525.954 /  
0x03400002

**Verwandte**

Siehe Befehle,

WinInfo(),

IntEdit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein IntEdit-Objekt hinweist.

**\_WinTypeInterval**

**Intervall-Objekt**

Wert 16.777.221 /  
0x01000005

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**Interval,**

**GanttGraph**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Interval**-Objekt eines **GanttGraph**-Objektes hinweist.



**\_WinTypeIvlBox**

**Box-Objekt**

Wert 16.777.224 /  
**0x01000008**

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**Box,**

**GanttGraph**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Box**-Objekt eines **GanttGraph**-Objektes hinweist.

\_WinTypeIvlLine

Linien-Objekt

Wert 16.777.225 /  
0x01000009

Verwandte

Befehle,

Siehe WinInfo(),

Line,

GanttGraph

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Line-Objekt eines GanttGraph-Objektes hinweist.

## Kontakt

**\_WinTypeLabel**

**Bezeichner-Objekt**

Wert 52.428.801 /  
0x03200001

**Verwandte**

Siehe Befehle,

WinInfo(),

Label

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Label-Objekt hinweist.

**\_WinTypeListColumn**

**Spalten-Objekt**

Wert 57.671.688 /  
0x03700008

**Verwandte**

**Befehle,**

**WinInfo(),**

Siehe **Column,**

**DataList,**

**RecList,**

**StoList**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Column**-Objekt eines **DataList**-, **RecList** - oder **StoList**-Objektes hinweist.

## Kontakt

**\_WinTypeMdiFrame**

**MDI-Fenster-Objekt**

Wert 51.380.227 /  
**0x03100003**

**Verwandte**

Siehe Befehle,

WinInfo(),

MdiFrame

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein MdiFrame-Objekt hinweist.

**\_WinTypeMenu**

**Menü-Objekt**

Wert 33.554.434 /  
0x02000002

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein Menü-Objekt hinweist.

## Kontakt

**\_WinTypeMenuButton**

**MenuButton-Objekt**

Wert 53.477.387 /

0x0330000B

**Verwandte**

Siehe Befehle,

WinInfo(),

MenuButton

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein MenuButton-Objekt hinweist.

**\_WinTypeMenuItem**

**Menüeintrag-Objekt**

Wert 33.554.435 /  
0x02000003

**Verwandte**

Siehe Befehle,

WinInfo(),

MenuItem

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein MenuItem-Objekt eines Menü-Objektes hinweist.



## Kontakt

**\_WinTypeMetaPicture**

**Meta-Grafik-Objekt**

**Wert 61.866.002 /**

**0x03B00012**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**MetaPicture**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein MetaPicture-Objekt hinweist.**

## Kontakt

**\_WinTypeNotebook**

**Notizbuch-Objekt**

**Wert 55.574.529 /  
0x03500001**

**Verwandte**

**Siehe Befehle,**

**WinInfo(),**

**Notebook**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Notebook-Objekt hinweist.**

**\_WinTypeNotebookPage**

**Notizbuchseiten-Objekt**

Wert 55.574.530 /  
0x03500002

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**NotebookPage,**

**Notebook**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **NotebookPage**-Objekt eines **Notebook**-Objektes hinweist.

## Kontakt

**\_WinTypePicture**

**Grafik-Objekt**

Wert 16.777.218 /  
0x01000002

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Picture**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Picture**-Objekt hinweist.

## Kontakt

\_WinTypePopupColor

Popup eines ColorButton- oder ColorEdit-Objekts

Wert 63.963.139 /

0x03d00003

Verwandte

Siehe Befehle,

WinInfo()

Rückgabewert von WinInfo() mit der Option \_WinType der auf das PopupColor-Objekt eines ColorButton- oder ColorEdit-Objekts hinweist.

**\_WinTypePopupList**

**PopupList-Objekt**

Wert 63.963.137 /  
0x03D00001

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**PopupList**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **PopupList**-Objekt eines Eingabe-Objekts hinweist.

**\_WinTypePpcObject**

**Druckprozessor-Objekt**

Wert 83.886.082 /  
0x05000002

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**ppcObject**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Druckprozessor-Objekt** hinweist.

## Kontakt

**\_WinTypePrintDevice**

**PrintDevice-Objekt**

Wert 67.108.866 /  
0x04000002

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**PrintDevice**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **PrintDevice**-Objekt hinweist.



## Kontakt

**\_WinTypePrinter**

**Printer-Objekt**

Wert 67.109.122 /  
0x04000102

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Printer**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Printer**-Objekt hinweist.

## Kontakt

**\_WinTypePrinterList**

**PrinterList-Objekt**

Wert 67.109.121 /  
0x04000101

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**PrinterList**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **PrinterList**-Objekt hinweist.

## Kontakt

**\_WinTypePrintJob**

**PrintJob-Objekt**

Wert 67.108.865 /  
0x04000001

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**PrintJob**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **PrintJob**-Objekt hinweist.

## Kontakt

**\_WinTypeProgress**

**Fortschrittsanzeigen-Objekt**

Wert 61.865.985 /

0x03B00001

**Verwandte**

Siehe Befehle,

WinInfo(),

Progress

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Progress-Objekt hinweist.

**\_WinTypePrtJobPreview**

**Druckvorschau-Objekt**

Wert 16.777.260 /  
0x0100002C

**Verwandte**

Siehe Befehle,

WinInfo(),

PrtJobPreview

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein PrtJobPreview-Objekt hinweist.

**\_WinTypePrtPreviewDlg**

**Druck-Vorschau-Container-Objekt**

Wert 16.777.264 /

**0x01000030**

**Verwandte**

Siehe Befehle,

WinInfo(),

PrtFrame

Rückgabewert von WinInfo() mit der Option \_WinType der auf das Control-Objekt der Druckvorschau hinweist.

**\_WinTypePrtPreviewDlg**

**Druckvorschau-Dialog**

Wert 65.011.745 /  
0x03E00021

**Verwandte**

Siehe Befehle,

WinInfo(),

PrtFrame

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Druckvorschau-Dialog hinweist. Dies dient zur Unterscheidung zwischen dem Vorschau-Objekt und dem Vorschau-Dialog.

**\_WinTypeRadioButton**

**Einfachauswahlknopf-Objekt**

Wert 53.477.379 /  
0x03300003

**Verwandte**

Siehe Befehle,

WinInfo(),

Radiobutton

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Radiobutton-Objekt hinweist.



## Kontakt

**\_WinTypeRecList**

**RecList-Objekt**

Wert 57.671.690 /  
0x0370000A

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**RecList**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **RecList**-Objekt hinweist.

## Kontakt

**\_WinTypeRecListPopup**

**RecListPopup-Objekt**

Wert 57.672.192 /  
0x03700200

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**RecListPopup**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **RecListPopup**-Objekt hinweist.

**\_WinTypeRecNavigator**

**Datensatznavigations-Objekt**

Wert 59.768.834 /  
0x03900002

**Verwandte**

Siehe Befehle,

WinInfo(),

RecNavigator

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein RecNavigator-Objekt hinweist.

## Kontakt

**\_WinTypeRecView**

**RecView-Objekt**

Wert 61.866.010 /  
**0x03B0001A**

**Verwandte**

Siehe Befehle,

WinInfo(),

RecView

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein RecView-Objekt hinweist.

**\_WinTypeRTFEdit**

**RTF-Eingabe-Objekt**

Wert 54.525.968 /  
**0x03400010**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**RtfEdit**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **RtfEdit**-Objekt hinweist.

**\_WinTypeScrollbar**

**Scrollbar-Objekt**

Wert 52.428.813 /  
0x0320000D

**Verwandte**

Siehe **Befehle**,

**WinInfo()**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf eine Scrollbar hinweist.

## Kontakt

**\_WinTypeScrollbox**

**Scrollbox-Objekt**

Wert 52.428.811 /  
0x0320000B

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Scrollbox**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Scrollbox**-Objekt hinweist.

**\_WinTypeSelDataObject**

**Selektions-Objekt der Mehrfachauswahl**

**Wert 117.440.513 /**

**0x07000001**

**Verwandte**

**Siehe Befehle,**

**WinInfo(),**

**SelectionData**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein SelectionData-Objekt hinweist.**



## Kontakt

**\_WinTypeStatusbar**

**Statusleisten-Objekt**

**Wert 60.817.668 /  
0x03A00104**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Statusbar**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Statusbar-Objekt hinweist.**

**\_WinTypeStatusBarButton**

**Statusleisten-Objekt**

Wert 60.817.669 /  
0x03A00105

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**Statusbar-Button,**

**Statusbar**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Statusbar-Button**-Objekt eines **Statusbar**-Objektes hinweist.

## Kontakt

**\_WinTypeStoList**

**StoList-Objekt**

Wert 57.672.705 /  
0x03700401

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**StoList**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **StoList**-Objekt hinweist.

## Kontakt

**\_WinTypeStoListPopup**

**StoListPopup-Objekt**

Wert 57.672.706 /  
0x03700402

**Verwandte**

Siehe Befehle,

WinInfo(),

StoListPopup

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein StoListPopup-Objekt hinweist.

## Kontakt

**\_WinTypeTextEdit**

**Texteingabe-Objekt**

Wert 54.525.961 /  
0x03400009

**Verwandte**

Siehe Befehle,

WinInfo(),

TextEdit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein TextEdit-Objekt hinweist.

## Kontakt

**\_WinTypeTimeEdit**

**Zeiteingabe-Objekt**

Wert 54.525.956 /  
0x03400004

**Verwandte**

Siehe Befehle,

WinInfo(),

TimeEdit

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein TimeEdit-Objekt hinweist.

## Kontakt

**\_WinTypeToolbar**

**Werkzeuggesteigen-Objekt**

**Wert 60.817.666 /  
0x03A00102**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Toolbar**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Toolbar-Objekt hinweist.**

**\_WinTypeToolbarButton**

**Werkzeuggesteigen-Schaltflächen-Objekt**

Wert 60.817.667 /

**0x03A00103**

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**Toolbar-Button,**

**Toolbar**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Toolbar-Button-Objekt** eines **Toolbar-Objektes** hinweist.



**\_WinTypeToolbarDock**

**Werkzeigleisten-Dock-Objekt**

Wert 60.817.665 /  
**0x03A00101**

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**ToolbarDock,**

**Toolbar**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **ToolbarDock**-Objekt für ein **Toolbar**-Objekt hinweist.

## Kontakt

**\_WinTypeToolbarMenu**

**Toolbar-Menü-Objekt**

**Wert 60.817.677 /**

**0x03A0010D**

**Verwandte**

**Siehe Befehle,**

**WinInfo()**

**Rückgabewert von WinInfo() mit der Option \_WinType der auf ein ToolbarMenu-Objekt hinweist.**

## Kontakt

**\_WinTypeToolbarRTF**

**RTF-Werkzeugleisten-Objekt**

Wert 60.817.674 /  
**0x03A0010A**

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**ToolbarRtf**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **ToolbarRtf**-Objekt hinweist.

## Kontakt

**\_WinTypeTrayFrame**

**Tray-Fenster-Objekt**

Wert 50.331.661 /  
0x0300000D

**Verwandte**

Siehe Befehle,

WinInfo(),

TrayFrame

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein TrayFrame-Objekt hinweist.

**\_WinTypeTreeNode**

**Knoten-Objekt**

Wert 58.720.261 /  
0x03800005

**Verwandte**

**Befehle,**

Siehe **WinInfo()**,

**TreeNode,**

**TreeView**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **TreeNode**-Objekt eines **TreeView**-Objektes hinweist.

## Kontakt

**\_WinTypeTreeView**

**Baum-Objekt**

Wert 58.720.259 /  
0x03800003

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**TreeView**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **TreeView**-Objekt hinweist.

## Kontakt

\_WinTypeWebNavigator

Dateianzeige-Objekt

Wert 59.768.835 /  
0x03900003

Verwandte

Siehe Befehle,

WinInfo(),

WebNavigator

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein WebcNavigator-Objekt hinweist.

**\_WinTypeWindowbar**

**Windowbar-Objekt**

Wert 60.817.673 /  
0x03A00109

**Verwandte**

Siehe Befehle,

WinInfo(),

Windowbar

Rückgabewert von WinInfo() mit der Option \_WinType der auf ein Windowbar-Objekt hinweist.



## Kontakt

**\_WinTypeChromium**

**Chromium-Objekt**

Wert 62.914.581 /  
0x03C00015

**Verwandte**

Siehe **Befehle**,

**WinInfo()**,

**Chromium**

Rückgabewert von **WinInfo()** mit der Option **\_WinType** der auf ein **Chromium-Objekt** hinweist.

Objektstatus

Objektstatus

Siehe Objektinformationsbefehle

- WinDialogMaximized
- WinDialogMinimized
- WinDialogNormal
- WinStateAttachClosed
- WinStateAttachMaximized
- WinStateAttachNormal

**\_WinDialogMaximized**

Fenster wird maximiert dargestellt

Wert 16 / 0x00000010

Verwandte

Befehle,

WinDialog(),

Siehe WinDialogRun(),

WinAdd(),

WinAddByName(),

WinUpdate(),

WinInfo()

Je nach Befehl hat **\_WinDialogMaximized** folgende Bedeutung:

- WinDialog(), WinDialogRun(), WinAdd() und WinAddByName() Dialog wird maximiert geladen.
- WinUpdate()  
Dialog wird zur Laufzeit maximiert. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.
- WinInfo()  
Liefert WinInfo( WinState) den Wert **\_WinDialogMaximized** zurück, ist der Dialog maximiert.

**\_WinDialogMinimized**

Fenster wird minimiert dargestellt

Wert 32 / 0x00000020

Verwandte

Befehle,

WinDialog(),

Siehe WinDialogRun(),

WinAdd(),

WinAddByName(),

WinUpdate(),

WinInfo()

Je nach Befehl hat **\_WinDialogMinimized** folgende Bedeutung:

- WinDialog(), WinDialogRun(), WinAdd() und WinAddByName() Dialog wird minimiert geladen.
- WinUpdate()  
Dialog wird zur Laufzeit minimiert. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.
- WinInfo()  
Liefert WinInfo(\_WinState) den Wert **\_WinDialogMinimized** zurück ist der Dialog minimiert.

**\_WinDialogNormal**

Fenster weder maximiert noch minimiert

Wert <sup>64</sup> /

0x00000040

Siehe **WinUpdate()** **WinInfo()**,

Je nach Befehl hat \_WinDialogNormal folgende Bedeutung:

- **WinDialog()** und **WinDialogRun()**

Der Dialog wird weder minimiert, noch maximiert angezeigt. Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

- **WinUpdate()**

Dialog wird zur Laufzeit von einem minimierten oder maximierten Status zurückgesetzt. Die Option kann nur angegeben werden, wenn **WinUpdState** als ersten Parameter übergeben wird.

- **WinInfo()**

Liefert **WinInfo( \_WinState)** den Wert \_WinDialogNormal zurück ist der Dialog weder maximiert noch minimiert.

**\_WinStateAttachClosed**

Objekt ist geschlossen

Wert 3

**Siehe** EvtAttachState,

\_WinState

Das GroupTile-Objekt ist/wird geschlossen.

**\_WinStateAttachMaximized**

**Maximierte Darstellung**

**Wert 2**

**Siehe** **EvtAttachState,**

**\_WinState**

**Das GroupTile-Objekt wird maximiert dargestellt.**

## Kontakt

**\_WinStateAttachNormal**

Normale Darstellung

Wert 1

**Siehe** EvtAttachState,  
\_WinState

Das GroupTile-Objekt ist/wird normal dargestellt.



## Oberflächenobjektbefehle

### Befehle für Oberflächenobjekte

Liste sortiert

nach

Gruppen,

Alphabetische

Liste aller

Siehe

### Befehle für Objekte:

- CtxDocEdit-Befehle
- DataList-Befehle
- GanttGraph-Befehle
- Menu-Befehle
- RtfEdit-Befehle
- Theme-Befehle
- TreeView-Befehle

### Befehle zum Laden/Anzeigen von Dialogen:

- WinAdd
- WinAddByName
- WinClose
- WinDialog
- WinDialogBox
- WinDialogResult
- WinDialogRun
- WinOpen
- WinSave
- WinUrmDialog

### Befehle zum Setzen/Ermitteln des Focus:

- WinFocusGet
- WinFocusSet

### Befehle für Ereignisse:

- ComEvtProcessGet
- ComEvtProcessSet
- WinEvtProcessGet
- WinEvtProcessSet
- WinEvtProcNameGet
- WinEvtProcNameSet

### Sonstige Befehle:

- ColorMake
- FontMake
- PointMake
- RangeMake

- RectMake
- RtfTabMake
- WinBarcodeSaveImage
- WinBeep
- WinBoxScrollVisible
- WinColorOpacityGet
- WinColorOpacitySet
- WinFlash
- WinHalt
- WinIconPreload
- WinLayer
- WinLstEdit
- WinPicSaveImage
- WinSearch
- WinSearchClear
- WinSearchPath
- WinSearchPathGet
- WinShutdownBlock
- WinUpdate
- WinWbnExecCommand

obj -> WinPropGet(int1, var2) : logic  Ermitteln einer Eigenschaft eines Oberflächenobjekts

obj        Objekt

int1       Eigenschaftsausprägung

var2       Variable

Resultat logic Ermittlungserfolg

Verwandte Befehle,

Siehe WinPropSet(),  
Eigenschaftsliste

Dieser Befehl liest eine Eigenschaft eines Oberflächenobjektes aus.

Als erster Parameter muss die Konstante der Eigenschaft übergeben werden. Die Konstanten setzen sich aus \_WinProp und dem Namen der Eigenschaft zusammen.

Im zweiten Parameter wird die Variable übergeben, in die der Wert der Eigenschaft kopiert werden soll.

Beispiel:

```
local{ tTitle        : alpha;}...// Auslesen des Objektitels $Obj->WinPropGet(_WinPropCaption, tTitl
```

Das Kommando kann ebenfalls dazu verwendet werden, um zu ermitteln, ob ein bestimmtes Objekt eine Eigenschaft besitzt. Ist eine Eigenschaft nicht vorhanden, liefert der Befehl den Wert false zurück.



Alternativ kann die Eigenschaft auch wie folgt ermittelt werden:













Beispiel:

```
local{ tTitle        : alpha;}...// Auslesen des Objektitels tTitle # $Obj->wpCaption;
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt (obj) ungültig

## Kontakt

`obj -> WinPropSet(int1, var2[, int3]) : logic`             Setzen einer Eigenschaft eines Oberflächenobjekts

**obj**      **Objekt**  
**int1**     **Eigenschaftsausprägung**  
**var2**     **Ausprägungswert**  
**int3**     **Flag**  
**Resultat** logic **Setzungserfolg**  
            Verwandte Befehle,  
**Siehe**    WinPropGet(),  
            Eigenschaftsliste

Dieser Befehl setzt eine Eigenschaft eines Oberflächenobjektes.

Als erster Parameter muss die Konstante der Eigenschaft übergeben werden. Die Konstanten setzen sich aus `_WinProp` und dem Namen der Eigenschaft zusammen.

Im zweiten Parameter wird der zu setzende Wert übergeben.

**Beispiel:**

```
// Setzen des Objektitels $Obj->WinPropSet(_WinPropCaption, 'New caption');
```



Alternativ kann die Eigenschaft auch wie folgt gesetzt werden:

**Beispiel:**

```
// Setzen des Objektitels $Obj->wpCaption # 'New caption';
```

Ein Vorteil der kompakteren Schreibweise gegenüber `WinPropSet()` und `WinPropGet()` besteht darin, dass schon während der Kompilierung eine Typprüfung vorgenommen wird:

```
$Obj->WinPropSet(_WinPropCaption, 100);
```

Diese Anweisung liefert während der Laufzeit den Rückgabewert false, da 100 vom Typ int ist.

```
$Obj->wpCaption # 100;
```

Dieser Befehl erzeugt bereits während der Kompilierung der entsprechenden Prozedur einen Fehler. Typ-Fehler dieser Art werden somit vermieden. Beim Lesen einer Eigenschaft, die im referenzierten Objekt nicht vorhanden ist, wird ein Laufzeitfehler erzeugt. Der Laufzeitfehler kann durch die Kapselung in einem try-Block unterbunden werden.

**Beispiel:**

```
try{ ErrTryCatch(_ErrPropInvalid, true); $Obj->wpCaptionInt # 100; ...}
```

Der optionale Parameter (int3) muss nur angegeben werden, wenn zusätzliche Informationen einer Eigenschaft zugeordnet werden können. Entsprechende Hinweise befinden sich in den Beschreibungen der Eigenschaften.

## Kontakt

### Beispiel:

```
// Aktuelle Notizbuchseite ohne Fokuswechsel setzentHdlNotebook->WinPropSet(_WinPropCurrent, tHdl
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Objekt (Obj) ungültig

obj -> PicExifInfo(var1, int2, int3, int4[, int5]) :



logic

Auslesen von Exif-Informationen aus JPEG-Bildern

obj      Deskriptor eines  
          Picture-Objekts

var1      Variable für die Information

int2      Index oder Tag-Id der

Information

int3      Modus (siehe Text)

int4      Verzeichnis

int5      Komponente (optional)

Resultat logic Funktion erfolgreich 

Siehe      Verwandte Befehle, Picture

Das Exchangeable Image File Format (Exif) speichert Informationen über das mit einer Digitalkamera aufgenommene Bild. Hierzu gehören beispielsweise Informationen wie die Marke des Aufnahmegerätes und die Modell-Bezeichnung. Die Exif-Informationen sind dabei Bestandteil eines Bildes im JPEG-Format. Das Format ist von der Japan Electronics and Information Technology Industries Association (JEITA) definiert. Nähere Informationen über das Format sind auch auf [www.exif.org](http://www.exif.org) nachzulesen.

Mit dem Befehl PicExifInfo() können die Informationen zu einem bestehenden JPEG ausgelesen werden. Dazu muss das Bild in einem Picture-Objekt geladen sein. Der Deskriptor des Objekts wird in (obj) übergeben. Nach dem Aufruf des Befehls steht in der in (var1) übergebenen Variable die gewünschte Information. Die Variable muss einen geeigneten Typ besitzen, um die entsprechende Information aufzunehmen.

In (int2) wird entweder ein Index oder eine Tag-Id übergeben. Die Tag-Ids bezeichnen bestimmte Exif-Informationen, es sind aber nur einige Tags standardisiert. Abhängig von der Quelle der Bilder, kann es zu Abweichungen sowohl bei den Tag-Ids, als auch bei den Informationen kommen. Bei Bildern aus unbekannter Quelle sollten zunächst die vorhandenen Informationen über eine Schleife ermittelt werden (siehe Beispiele).

In (int3) wird der Modus übergeben. Über den Modus wird bestimmt, welche Informationen zu dem in (int2) angegebenen Index oder angegebener Tag-Id zurückgegeben werden. Folgende Konstanten stehen zur Verfügung:

<u>PicExifTagName</u>	Name des Tags
<u>PicExifTagTitle</u>	Tag-Titel
<u>PicExifTagDescription</u>	Tag-Beschreibung
<u>PicExifTagValue</u>	Tag-Wert
<u>PicExifTagType</u>	Tag-Typ (siehe <u>Type...</u> )
<u>PicExifTagComponents</u>	Anzahl der Komponenten bei zusammengesetzten Datentypen

Jede der Konstanten kann mit PicExifTagIndex kombiniert werden, um anzuzeigen, dass in (int2) ein Index und keine Tag-Id angegeben wurde. Bei der Ermittlung des Datentyps (PicExifTagType) wird ein Wert zurückgegeben, der mit den Type...-Konstanten verglichen werden kann. Handelt es sich um einen zusammengesetzten Datentyp, wird TypeOther zurückgegeben. Die Datentypen der

## Kontakt

einzelnen Komponenten müssen dann durch erneute Aufrufe der Anweisung ermittelt werden. Die Anzahl der Komponenten kann mit PicExifTagComponents ermittelt werden. Die Nummer wird dann in (int5) angegeben.

Die Informationen innerhalb des JPEG sind in Verzeichnissen (IFD, Image File Directory) organisiert. Wird direkt über eine Tag-Id auf eine Information zugegriffen, wird als Verzeichnis -1 angegeben. Beim Zugriff über einen Index, muss das Verzeichnis angegeben werden. Folgende Konstanten können in (int4) übergeben werden:

<b>-1</b>	<b>Zugriff über eine Tag-Id</b>
<b><u>PicExifIfd0</u></b>	<b>Enthält unter anderem Informationen über die Kamera</b>
<b><u>PicExifIfd1</u></b>	<b>Enthält unter anderem Informationen zur Bild-Auflösung</b>
<b><u>PicExifIfdExif</u></b>	<b>Enthält unter anderem Informationen zur Exif-Version</b>
<b><u>PicExifIfdGps</u></b>	<b>Enthält (GPS-)Informationen über den Standort</b>
<b><u>PicExifIfdInterop</u></b>	<b>Enthält Informationen zu den Interoperabilitätsregeln</b>

Die Anweisung konnte erfolgreich ausgeführt werden, wenn true zurückgegeben wurde. false wird zum Beispiel dann zurückgegeben, wenn eine angegebene Tag-Id nicht existiert.

Beispiele:

```
// Ermitteln des ersten Tag-Namens tTagIndex # 1; tResult # aPicture->PicExifInfo(tTagName, tTagInd
```

In der Beispiel-Datenbank CodeLibrary befindet sich ein ausführliches Beispiel.

Mögliche Laufzeitfehler:

<b><u>ErrHdlInvalid</u></b>	<b>Der in (obj) übergebene Deskriptor ist ungültig</b>
<b><u>ErrValueInvalid</u></b>	<b>In (int3) oder (int4) wurde eine ungültige Konstante übergeben.</b>
<b><u>ErrFldType</u></b>	<b>Die in (var1) übergebene Variable hat den falschen Datentyp</b>

### Barcode-Befehle

Befehle zum Verwenden eines Barcode-Objektes

Verwandte

Befehle,

Siehe Barcode,

Befehlsgruppen,

Befehlsliste

Befehle

- WinBarcodeSaveImage





## Kontakt

ErrValueInvalid Ungültiges Format (int2) angegeben.

## Kontakt

Befehle für dynamische Objekte








Befehle zur dynamischen Erstellung und Zerstörung von Objekten Verwandte

Siehe Befehle,  
Befehlsgruppen,  
Befehlsliste

Befehle

- WinAdd
- WinCopy
- WinCreate
- WinDestroy
- WinRemove

## Kontakt

obj -> WinAdd(handle1[, int2[, handle3]]) : int        Oberflächenobjekt in  
ein anderes Oberflächenobjekt laden

obj Elternojekt

handle1 einzufügendes Oberflächenobjekt

Optionen (optional)

WinAddHidden

MDI-Fenster

unsichtbar

laden

WinDialogMaximized MDI-Fenster

int2

maximiert

laden

WinDialogMinimized MDI-Fenster

minimiert

laden

handle3 Deskriptor des nachfolgenden

Objektes (optional)

Laderesultat

**Resultat** int

ErrOk Laden erfolgreich



Verwandte Befehle,

WinAddByName(), WinCreate(),

Siehe

WinRemove(), WinDestroy(),

Ereignisabläufe MdiFrame

Der Befehl fügt dem angegebenen Elternojekt (obj) das Oberflächenobjekt (handle1) hinzu. Bei dem Oberflächenobjekt muss es sich um einen MdiFrame oder um ein mit WinCreate() angelegtes Objekt handeln. Das Objekt (handle1) kann bereits weitere mit WinCreate() und WinAdd() untergeordnete Oberflächenobjekte enthalten.

Ist das Objekt (handle1) ein MdiFrame, kann über den optionalen Parameter (int2) mit der Konstanten WinAddHidden angegeben werden, dass das MdiFrame-Objekt unsichtbar geladen werden soll. In diesem Fall muss das MdiFrame-Objekt später mit dem Befehl WinUpdate() mit dem Parameter WinUpdOn gezeichnet werden. Mit den Optionen WinDialogMaximized bzw. WinDialogMinimized kann der MdiFrame maximiert oder minimiert werden.

Sind dem MDI-Fenster über die Eigenschaft DbRecBuf eigene Feldpuffer zugeordnet worden, stehen diese nach diesem Befehl zur Verfügung und können initialisiert werden.

Wurde das Oberflächenobjekt (handle1) mit WinCreate() erstellt, kann das nachfolgende Objekt im Argument (handle3) angegeben und somit die Objektreihenfolge definiert werden.



Das Objekt (handle1) darf zuvor nicht bereits mit WinAdd() zu einem anderen Objekt hinzugefügt werden, außer es wurde anschließend mit WinRemove() wieder entfernt.

Bei dem Oberflächenobjekt (handle1) kann es sich nicht um ein Frame-, AppFrame-oder TrayFrame-Objekt handeln. Um diese zu starten, muss, auch für dynamisch erstellte, weiterhin der Befehl WinDialogRun() verwendet werden.

## Kontakt

Wird das Elternobjekt (obj) bereits angezeigt, erfolgt auch die Darstellung des hinzugefügten Oberflächenobjektes (handle1) direkt nach der Ausführung des Befehls.

Als Resultat kann der Fehlerwert ErrType zurückgegeben werden, wenn das hinzuzufügende Objekt (handle1) nicht in das Elternobjekt (obj) eingefügt werden kann. Das Resultat ist ErrExists, wenn das hinzuzufügende Objekt (handle1) bereits einem Objekt hinzugefügt wurde. Konnte ein MdiFrame nicht zu einem AppFrame hinzugefügt werden, ist das Resultat ErrOutOfMemory. Das Resultat ist ErrUnavailable, wenn eine Spalte vor einer bestehenden Spalte eingefügt wird und die Liste bereits Inhalt besitzt. Ist kein Fehler aufgetreten, wird ErrOk zurückgegeben.

### Hinweise für Spalten-Objekte

Dynamische Spalten-Objekte können auch einer nicht dynamisch erstellten DataList oder RecList hinzugefügt werden. Diese darf auch bereits nicht dynamisch erstellte Spalten enthalten.

Ein Spalten-Objekt besitzt eine Anzeigeposition und eine Indexposition. Die Anzeigeposition definiert, wo die Spalte innerhalb der Liste angezeigt wird und kann sich z. B. durch Benutzerinteraktionen mit der Spalte (z. B. Verschieben der Spalte durch den Anwender) ändern. Die Anzeigeposition kann durch die Eigenschaft ClnOrder gesetzt oder auch abgefragt werden.

Die Indexposition definiert, welche Position die Zelle im Datensatz (Zeile) besitzt. Diese Position wird den Befehlen WinLstCellSet() und WinLstCellGet() übergeben, wenn die Daten einer Zelle gesetzt oder abgefragt werden sollen. Die Indexposition kann mit dem Befehl WinInfo() und der Option WinItem ermittelt werden. Bei WinAdd() unter Angabe einer nachfolgenden Spalte erhält die neu hinzugefügte Spalte die Indexposition der nachfolgenden Spalte. Die Anzeigeposition wird über die Eigenschaft ClnOrder definiert.



Die Angabe eines nachfolgenden Spalten-Objektes ist nicht zulässig, wenn die Liste bereits einen Inhalt besitzt (Rückgabewert ErrUnavailable). Es können jedoch Spalten am Ende eingefügt werden (WinAdd() ohne nachfolgendes Objekt).

Nachdem ein Spalten-Objekt einer DataList hinzugefügt wurde, hat diese zunächst keinen Inhalt. Mit dem Befehl WinLstCellSet() kann der Spalteninhalt entsprechend des durch ClnType definierten Spalten-Typs gesetzt werden. Der Befehl WinLstCellGet() liefert den Wert false, wenn die Zelle der Spalte noch keinen Inhalt besitzt.

### Hinweise für GroupColumn-Objekte

Dynamische GroupColumn-Objekte können auch einem nicht dynamisch erstellten RecView- oder einem nicht dynamisch erstellten, übergeordneten GroupColumn-Objekt hinzugefügt werden.

Ein GroupColumn-Objekt besitzt eine Anzeige- und eine Indexposition. Die Anzeigeposition definiert, wo das GroupColumn-Objekt im RecView bzw. im

## Kontakt

übergeordneten GroupColumn-Objekt angezeigt wird. Die Anzeigeposition kann durch die Eigenschaft VisibleOrder gesetzt und abgefragt werden.

Die Indexposition definiert eine eindeutige fortlaufende Nummer für die Eigenschaft SelectorItem bzw. SelectorSubItem.

Die Angabe eines nachfolgenden GroupColumn-Objektes ist nicht zulässig, wenn das RecView bereits einen Inhalt besitzt. Es können jedoch GroupColumn-Objekte am Ende eingefügt werden (WinAdd() ohne nachfolgendes Objekt).

Wird das RecView-Objekt angezeigt, während WinAdd() für ein GroupColumn-Objekt durchgeführt wird, dann hat dies zur Folge, dass das Ereignis EvtLstGroupInit aufgerufen wird.

### Hinweise zum PopupList-Objekt

Das Objekt kann den Eingabeobjekten (Edit, IntEdit, ...) hinzugefügt werden. Dem PopupList-Objekt kann wiederum ein DataListPopup, RecListPopup- oder StoListPopup-Objekt hinzugefügt werden.

Einem GroupColumn-Objekt kann ein weiteres GroupColumn-Objekt untergeordnet werden. Dies ist jedoch nur zulässig, wenn das übergeordnete Objekt nicht bereits einem GroupColumn-Objekt untergeordnet ist und das unterzuordnende GroupColumn-Objekt seinerseits keine untergeordneten GroupColumn-Objekte enthält.

### Beispiele:

```
// MDI-Frame 'Addresses' laden tMdiFrame # WinOpen('Addresses');// Wenn hinzufügen des MDI-Fenste
```

### Mögliche Laufzeitfehler:

#### ErrHdlInvalid

Elternobjekt (obj) oder hinzuzufügendes Oberflächenobjekt (handle1) ungültig oder das Oberflächenobjekt (handle1) ist nicht dynamisch erstellt worden.

#### ErrMemExhausted

Fenstererstellung ist fehlgeschlagen.  
Das nachfolgende Objekt (handle3) ist angegeben, jedoch kein Oberflächenobjekt oder kein Kindobjekt von dem angegebenen Elternobjekt (obj).

#### ErrIllegalOp

Das hinzuzufügende Objekt (handle1) ist ein Frame, MdiFrame oder AppFrame und ein nachfolgendes Objekt (handle3) ist angegeben.

## Kontakt

obj -> WinCopy([int1]) :



handle

Oberflächenobjekt kopieren

obj Oberflächenobjekt

Optionen (optional)

WinCopyDefault Alle Eigenschaften

und Ereignisse

werden kopiert

int1

(Standard).

WinCopyNoEvents Ereignisfunktionen

werden nicht

kopiert.

Deskriptor des neuen

Oberflächenobjektes oder

Fehlerwert:

ErrType Das

Resultat handle Oberflächenobjekt,

oder eines der

untergeordneten

Objekte kann nicht

kopiert werden.

Siehe Verwandte Befehle, WinCreate(),

WinAdd()

Mit diesem Befehl wird das Oberflächenobjekt (obj) inklusive der untergeordneten Oberflächenobjekte kopiert. Hierbei werden alle Eigenschaften und Ereignisse ebenfalls übernommen.

Bei (obj) kann es sich sowohl um ein dynamisch erstelltes (WinCreate()) als auch um ein aus der Datenbank geladenes Objekt handeln (WinOpen()). Sofern es sich um ein Objekt aus der Datenbank handelt, muss es sich bei (obj) und all seinen untergeordneten Objekten um dynamisch erstellbare Objekte handeln, also um solche Objekte, die auch mit WinCreate() erstellt werden können.

Folgende Optionen können im optionalen Argument (int1) angegeben werden:

WinCopyDefault Alle Eigenschaften und Ereignisse werden kopiert (Standard).

WinCopyNoEvents Ereignisfunktionen werden nicht kopiert.

Um das kopierte Oberflächenobjekt anzuzeigen, muss es einem sichtbaren Oberflächenobjekt mit WinAdd() hinzugefügt werden.

Konnte das Objekt kopiert werden, wird der Deskriptor auf das neue Objekt zurückgegeben, andernfalls der Fehlerwert ErrType.

Beispiele:

```
// Groupbox inklusive Ereignisse kopierenNewObj # $grpGroup->WinCopy();if (tNewObj > 0) tFrame-
```

Mögliche Laufzeitfehler:

## Kontakt

ErrHdlInvalid

Oberflächenobjekt (obj) enthält keinen gültigen Deskriptor oder

verweist nicht auf ein dynamisch erstellbares Objekt.

ErrValueInvalid Bei (int1) wurde eine ungültige Option angegeben.



## Kontakt

**WinCreate(int1[, alpha2[,  
alpha3[, handle4[,  
handle5]]]]) : handle**  
Oberflächenobjekt erzeugen  
**int1** Objekttyp  
**alpha2** Name des Objektes (optional)  
**alpha3** Caption des Objektes (optional)  
**handle4** Deskriptor des Elternobjektes  
(optional)  
**handle5** Deskriptor des nachfolgenden  
Objektes (optional)  
Objekt-Deskriptor oder

### Resultat **handle Fehlerwert**

Siehe Verwandte Befehle, WinAdd(),  
WinRemove(), WinDestroy()

Der Befehl erstellt ein Oberflächenobjekt vom Typ (int1) und liefert den Deskriptor auf dieses zurück. Optional können sowohl der Name im Parameter (alpha2), als auch die Caption im Parameter (alpha3) angegeben werden. Die Caption (alpha3) wird nur ausgewertet, wenn das entsprechende Objekt diese Eigenschaft besitzt.

Wird das Elternobjekt (handle4) angegeben, wird das erstellte Objekt nach dem Setzen der Eigenschaften Name und Caption unmittelbar an WinAdd() übergeben. Das nachfolgende Objekt (handle5) wird nur ausgewertet, wenn das Elternobjekt (handle4) angegeben ist.



Um das erstellte Objekt anzuzeigen muss es einem sichtbaren Oberflächenobjekt durch Angabe eines Elternobjektes (handle4) oder durch Übergabe an WinAdd() hinzugefügt werden.

Das neu erstellte Objekt kann nicht über das with-Statement angesprochen werden.

Die Suche über WinSearch() ist nur möglich, wenn das Objekt einem anderen Oberflächenobjekt untergeordnet ist. Vorzugsweise sollte der zurückgelieferte Deskriptor verwendet werden.

Für die Ausführung von dynamisch erzeugten Frame- und AppFrame-Objekte muss weiterhin der Befehl WinDialogRun() verwendet werden. Zum Schließen eines angezeigten Fensters kann wahlweise WinClose(), WinRemove() oder WinDestroy() benutzt werden. Um das nicht mehr angezeigte Fenster zu zerstören, stehen WinClose() und WinDestroy() zur Verfügung.

Als Resultat kann der Fehlerwert ErrType zurückgegeben werden, wenn ein Objekt von dem Typ (int1) nicht angelegt, oder nicht in das Elternobjekt (handle4) eingefügt werden kann. Ist kein Fehler aufgetreten, wird der Deskriptor des angelegten Objektes zurückgegeben.

### Beispiele:

```
// NotebookPage zu einem Notebook-Objekt hinzufügenWinCreate(_WinTypeNotebookPage, 'nbpView', 'An
```

### Unicode

## Kontakt

Dynamisch erstellte Objekte können auch Unicode-Objekte sein. Die Festlegung, ob ein Objekt Unicode ist oder nicht, wird bei WinCreate bzw. WinAdd() getroffen.

Sofern der Deskriptor des Elternobjektes bei WinCreate angegeben ist, wird ein Unicode-Objekt erstellt, sofern es sich beim Elternobjekt um ein Unicode-Objekt handelt.

Wird ein dynamisch erstelltes Objekt per WinAdd() zu einem Unicode-Objekt hinzugefügt, dann wird das dynamisch erstellte Objekt ebenfalls Unicode.

Bestimmte Eigenschaften eines Objektes können Unicode-Eigenschaften sein. Bei dynamisch erstellten Unicode-Objekten wird der UTF-8-Zeichensatz erwartet.

Beispiele:

```
myLabel # WinCreate(_WinTypeLabel);myLabel->wpCaption # CaptionString; // Objekt nicht Unicode. D
myLabel # WinCreate(_WinTypeLabel,'myLabel',CaptionString,UnicodeFrame); // myLabel wird als Unic
```

Folgende Objekte können dynamisch erzeugt werden:

Palette	Objekt	Objekttyp
Form	<u>Frame</u>	<u>_WinTypeFrame / _WinTypeDialog</u>
Form	<u>AppFrame</u>	<u>_WinTypeAppFrame</u>
Form	<u>MdiFrame</u>	<u>_WinTypeMdiFrame</u>
Eingabe	<u>Edit</u>	<u>_WinTypeEdit</u>
Eingabe	<u>IntEdit</u>	<u>_WinTypeIntEdit</u>
Eingabe	<u>BigIntEdit</u>	<u>_WinTypeBigIntEdit</u>
Eingabe	<u>FloatEdit</u>	<u>_WinTypeFloatEdit</u>
Eingabe	<u>DecimalEdit</u>	<u>_WinTypeDecimalEdit</u>
Eingabe	<u>TimeEdit</u>	<u>_WinTypeTimeEdit</u>
Eingabe	<u>DateEdit</u>	<u>_WinTypeDateEdit</u>
Eingabe	<u>ColorEdit</u>	<u>_WinTypeColorEdit</u>
Eingabe	<u>FontNameEdit</u>	<u>_WinTypeFontNameEdit</u>
Eingabe	<u>FontSizeEdit</u>	<u>_WinTypeFontSizeEdit</u>
Eingabe	<u>TextEdit</u>	<u>_WinTypeTextEdit</u>
Eingabe	<u>RtfEdit</u>	<u>_WinTypeRTFEdit</u>
Schaltflächen	<u>Button</u>	<u>_WinTypeButton</u>
Schaltflächen	<u>MenuButton</u>	<u>_WinTypeMenuButton</u>
Schaltflächen	<u>ColorButton</u>	<u>_WinTypeColorButton</u>
Schaltflächen	<u>Radiobutton</u>	<u>_WinTypeRadioButton</u>
Schaltflächen	<u>Checkbox</u>	<u>_WinTypeCheckBox</u>
Ansicht	<u>DataList</u>	<u>_WinTypeDataList</u>

## Kontakt

Ansicht	<u>DataListPopup</u>	<u>_WinTypeDataListPopup</u>
Ansicht	<u>StoList</u>	<u>_WinTypeStoList</u>
Ansicht	<u>StoListPopup</u>	<u>_WinTypeStoListPopup</u>
Ansicht	<u>RecList</u>	<u>_WinTypeRecList</u>
Ansicht	<u>RecListPopup</u>	<u>_WinTypeRecListPopup</u>
Ansicht	<u>RecView</u>	<u>_WinTypeRecView</u>
Ansicht	<u>TreeView</u>	<u>_WinTypeTreeView</u>
Ausgabe	<u>Label</u>	<u>_WinTypeLabel</u>
Ausgabe	<u>Icon</u>	<u>_WinTypeIcon</u>
Ausgabe	<u>Picture</u>	<u>_WinTypePicture</u>
Ausgabe	<u>Overlay (Picture)</u>	<u>_WinTypePicture</u>
Ausgabe	<u>MetaPicture</u>	<u>_WinTypeMetaPicture</u>
Ausgabe	<u>Overlay (MetaPicture)</u>	<u>_WinTypeMetaPicture</u>
Ausgabe	<u>Animation</u>	<u>_WinTypeAnimation</u>
Ausgabe	<u>DocView</u>	<u>_WinTypeDocView</u>
Ausgabe	<u>WebNavigator</u>	<u>_WinTypeWebNavigator</u>
Ausgabe	<u>Chromium</u>	<u>_WinTypeChromium</u>
Ausgabe	<u>PrtJobPreview</u>	<u>_WinTypePrtJobPreview</u>
Toolbar	<u>Toolbar-Button</u>	<u>_WinTypeToolbarButton</u>
Toolbar	<u>Statusbar-Button</u>	<u>_WinTypeStatusbarButton</u>
Anordnung	<u>Groupbox</u>	<u>_WinTypeGroupBox</u>
Anordnung	<u>Notebook</u>	<u>_WinTypeNotebook</u>
Anordnung	<u>NotebookPage</u>	<u>_WinTypeNotebookPage</u>
Anordnung	<u>GroupSplit</u>	<u>_WinTypeGroupSplit</u>
Anordnung	<u>GroupTile</u>	<u>_WinTypeGroupTile</u>
Anordnung	<u>Scrollbox</u>	<u>_WinTypeScrollbox</u>
Anordnung	<u>Divider</u>	<u>_WinTypeDivider</u>
COM	<u>CtxOffice</u>	<u>_WinTypeCtxOffice</u>
COM	<u>CtxAdobeReader</u>	<u>_WinTypeCtxAdobeReader</u>
COM	<u>CtxDocEdit</u>	<u>_WinTypeCtxDocEdit</u>
Weitere	<u>Hyperlink</u>	<u>_WinTypeHyperLink</u>
Weitere	<u>Calendar</u>	<u>_WinTypeCalendar</u>
Weitere	<u>Progress</u>	<u>_WinTypeProgress</u>
Weitere	<u>RecNavigator</u>	<u>_WinTypeRecNavigator</u>
	<u>PopupList</u>	<u>_WinTypePopupList</u>
	<u>Column</u>	<u>_WinTypeListColumn</u>

## Kontakt

GroupColumn

WinTypeGroupColumn

GroupTile-Button WinTypeGroupTileButton Besonderheiten

bei der Erstellung dynamischer Spalten- und GroupColumn-Objekte:

Eine dynamisch angelegte Spalte ( WinTypeListColumn ) hat standardmäßig folgende Eigenschaften:

- Sie ist vom Typ alpha ( ClmType = TypeAlpha )
- Sie ist sichtbar ( Visible = true )
- Spaltenbreite ist 70 Pixel ( ClmWidth = 70 )
- ClmOrder steht auf dem höchstmöglichen Wert ( MaxInt ), so dass ohne weitere Veränderung der Eigenschaft ein Hinzufügen der Spalte am Ende erfolgt (Anzeigeposition).

Die Eigenschaft ClmType kann nur gesetzt werden, wenn die Spalte noch keiner Liste hinzugefügt wurde, oder die Liste leer ist.

Die Spalte kann nur vor einer bestehenden Spalte (handle5) eingefügt werden, wenn die Liste leer ist. Andernfalls ist der Rückgabewert ErrUnavailable. Am Ende einer Liste können Spalten immer eingefügt werden.

Eine dynamisch angelegte GroupColumn ( WinTypeGroupColumn ) hat standardmäßig folgende Eigenschaften:

- ContentType ist WinContentTypeText
- ContentSource ist WinContentSourceField
- ContentHeightMax = 0 (keine Höhenbegrenzung)
- WordBreak = false
- AreaMarginLeft, AreaMarginTop, AreaMarginRight und AreaMarginBottom sind auf 4 Pixel eingestellt.
- BorderWidth ist 1 und BorderType ist WinBorderTypeNone
- Sie ist sichtbar ( Visible = true )
- Spaltenbreite ist 70 Pixel ( ClmWidth = 70 )
- VisibleOrder steht auf 0. Die Eigenschaft kann erst nach dem Hinzufügen der Spalte zum RecView bzw. zur übergeordneten Spalte gesetzt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor des nächsten Objektes (handle5) ist angegeben, jedoch kein Elternobjekt (handle4).

Anlegen des Objektes ist fehlgeschlagen. Dies kann zum Beispiel ErrMemExhausted auftreten, wenn die maximale Anzahl der Oberflächenobjekte von 10.000 erreicht ist.

ErrValueInvalid Name (alpha2) ist nicht leer und enthält keinen gültigen

Objektnamen. (Beschränkungen des Namens siehe Name)

ErrIllegalOp Das nachfolgende Objekt (handle5) ist angegeben, jedoch kein Oberflächenobjekt oder kein Kindobjekt von dem angegebenen Elternobjekt (handle4).

## Kontakt

Das angelegte Objekt (int1) ist ein Frame, MdiFrame oder AppFrame und ein nachfolgendes Objekt (handle5) ist angegeben.

Die Caption (alpha3) des neuen Objektes konnte nicht gesetzt werden.  
Der Fehler tritt auf, wenn es zwar eine

ErrPropInvalid

Caption-Eigenschaft beim Objekt gibt, diese jedoch nicht gesetzt werden kann (z. B. weil die Eigenschaft read-only ist).

obj -> WinDestroy([logic1])  Frame- oder

Oberflächenobjekt zerstören

zu zerstörendes

obj

Oberflächenobjekt

nur untergeordnete

logic1 Objekte zerstören

(optional)

Verwandte Befehle,

WinCreate(),

Siehe

WinAdd(),

WinRemove()

Der Befehl zerstört ein durch WinCreate() erstelltes Oberflächenobjekt. Das Objekt wird im Argument (obj) angegeben. Sollte das Objekt noch nicht mit WinRemove() vom Elternobjekt gelöst worden sein, geschieht dies hier automatisch.

Der Deskriptor (obj) ist nach der Zerstörung nicht mehr gültig.

Enthält das Objekt (obj) untergeordnete Objekte, werden diese ebenfalls zerstört.

Dynamisch erstellte Frame-Objekte können alternativ auch mit WinClose() geschlossen und zerstört werden. Dabei ist zu beachten, dass ein angezeigtes Fenster beim ersten Aufruf von WinClose() nur geschlossen wird. Der Befehl WinDestroy() muss hingegen nur einmal aufgerufen werden.

Wird im Argument (logic1) true übergeben, werden nur die untergeordneten dynamisch erstellten Oberflächenobjekte entfernt und zerstört. In diesem Fall wird das Objekt (obj) weder entfernt noch zerstört.



Der übergebene Deskriptor (obj) muss mit WinCreate() erstellt worden sein.

Hinweise zum Entfernen von Spalten-Objekten:

Wenn die Liste Daten (Zeilen) enthält, kann immer nur die zuletzt hinzugefügte Spalte (mit der größten Indexposition) entfernt bzw. zerstört werden. Die Angabe einer anderen Spalte führt zu dem Laufzeitfehler ErrIllegalOp. Sofern die Liste keine Daten enthält, kann eine beliebige dynamisch erstellte Spalte entfernt werden.

Hinweise zum Entfernen von GroupColumn-Objekten:

Wenn das RecView Gruppen enthält, kann immer nur das zuletzt hinzugefügte GroupColumn-Objekt (mit der größten Indexposition) entfernt bzw. zerstört werden. Dies gilt auch bei untergeordneten GroupColumn-Objekten. Sofern das RecView leer ist (also keine Gruppen enthält), kann ein beliebiges GroupColumn-Objekt entfernt werden.

Beispiel:

```
sub EvtClicked( aEvt : event; // Ereignis): logic; local { tDlg
```

Mögliche Laufzeitfehler:

## Kontakt

**ErrHdlInvalid** Das Oberflächenobjekt (obj) ist ungültig oder wurde nicht mit WinCreate() dynamisch erstellt.

obj ->



**WinRemove()**

**Objekt entfernen**  
zu entfernendes

obj

**Oberflächenobjekt**

**Verwandte Befehle,**

Siehe WinCreate(),

WinAdd(),

WinDestroy()

Dieser Befehl entfernt ein mit WinAdd() hinzugefügtes Oberflächenobjekt (obj) aus seinem Elternobjekt.

Enthält das entfernte Objekt untergeordnete Objekte, werden diese mit entfernt.

Ein entferntes Objekt kann mit WinAdd() wieder einem Oberflächenobjekt hinzugefügt, oder mit WinDestroy() zerstört werden.

Wird als (obj) ein Fenster-Objekt angegeben, wird es geschlossen. Es kann anschließend mit WinClose() oder WinDestroy() zerstört werden.



Der übergebene Deskriptor (obj) muss mit WinCreate() erstellt worden sein.  
**Hinweise zum Entfernen von Spalten-Objekten:**

Wenn die Liste Daten (Zeilen) enthält, kann immer nur die zuletzt hinzugefügte Spalte (mit der größten Indexposition) entfernt bzw. zerstört werden. Die Angabe einer anderen Spalte führt zu dem Laufzeitfehler ErrIllegalOp. Sofern die Liste keine Daten enthält, kann eine beliebige dynamisch erstellte Spalte entfernt werden.

**Hinweise zum Entfernen von GroupColumn-Objekten:**

Wenn das RecView Gruppen enthält, kann immer nur das zuletzt hinzugefügte GroupColumn-Objekt (mit der größten Indexposition) entfernt bzw. zerstört werden. Dies gilt auch bei untergeordneten GroupColumn-Objekten. Sofern das RecView leer ist (also keine Gruppen enthält), kann ein beliebiges GroupColumn-Objekt entfernt werden.

**Beispiel:**

```
sub EvtClicked( aEvt : event; // Ereignis): logic; local { tDlg
```

**Mögliche Laufzeitfehler:**

**ErrHdlInvalid**

Das Oberflächenobjekt (obj) ist ungültig oder wurde nicht mit WinCreate() dynamisch erstellt.

**ErrIllegalOp**

Das Objekt (obj) besitzt kein Elternobjekt oder das Objekt ist eine Spalte, die sich nicht am Ende einer gefüllten Liste befindet.



## Chromium-Befehle

Befehle und Konstanten für Chromium-Objekte Liste  
sortiert  
nach  
Gruppen,

Siehe

Alphabetische  
Liste aller

## Befehle

- WinCroNavigate
- WinCroPrint
- WinCroReload
- WinCroSelection

## Konstanten

- WinCroDevToolsSideBottom
- WinCroDevToolsSideLeft
- WinCroDevToolsSideNone
- WinCroDevToolsSideRight
- WinCroDevToolsSideTop
- WinCroNavBack
- WinCroNavForward
- WinCroReloadIgnoreCache
- WinCroReloadNormal
- WinCroSelCopy
- WinCroSelCut
- WinCroSelDelete
- WinCroSelPaste
- WinCroSelSelectAll

obj ->



**WinCroNavigate(int1)**

**Navigiert zu einer URL**

Objekt  
obj

**(Chromium-Objekt)**

int1    Art der Navigation

Siehe Verwandte Befehle

Navigiert zu einer URL, deren Inhalt im Verlauf der Navigation bereits angezeigt wurde, falls eine solche Navigation stattgefunden hat.

Im Parameter (int1) können folgende Optionen übergeben werden:

- **\_WinCroNavBack (1)**

Navigiert zur vorhergehenden Seite.

- **\_WinCroNavForward (2)**

Navigiert zur nächsten Seite.

**Mögliche Laufzeitfehler**

**\_ErrHdlInvalid**    Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

**\_ErrValueInvalid**    In Argument (int1) wurde keiner der gültigen Werte für die Navigation angegeben.

## Kontakt

obj ->



**WinCroPrint()**

**Inhalt drucken**

Objekt

obj

**(Chromium-Objekt)**

Siehe Verwandte Befehle

Der Befehl ruft einen Dialog zum Drucken des Inhaltes der aktuell angezeigten Seite aus.

**Mögliche Laufzeitfehler**

**ErrHdlInvalid** Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

## Kontakt

obj -> WinCroReload(int1)



Lädt den Inhalt der aktuellen URL erneut

obj  
Objekt

(Chromium-Objekt)

int1 Optionen

Siehe Verwandte Befehle

Der Befehl lädt den Inhalt der aktuell angezeigten Seite erneut. Über Optionen kann gesteuert werden, ob die Seite neu angefordert oder aus dem Cache geladen werden soll.

Im Parameter (int1) können folgende Optionen übergeben werden:

- \_WinCroReloadIgnoreCache (1)

Zwischengespeicherte Inhalte werden erneut angefordert.

- \_WinCroReloadNormal (0)

Zwischengespeicherte Inhalte werden nicht erneut angefordert (default).

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

ErrValueInvalid In Argument (int1) wurde keine der gültigen Optionen angegeben.

## Kontakt

**obj -> WinCroSelection(int1) Aktionen für  
den ausgewählten Inhalt**



obj      Objekt

          (Chromium-Objekt)

int1    Aktion

Siehe Verwandte Befehle

Der Befehl führt Aktionen auf den aktuell angezeigten Inhalt aus, sofern möglich.

Im Parameter (int1) können folgende Optionen übergeben werden:

- \_WinCroSelCopy (1)

Kopiert den ausgewählten Inhalte in die Zwischenablage.

- \_WinCroSelCut (2)

Schneidet den ausgewählten Inhalt in die Zwischenablage aus.

- \_WinCroSelDelete (4)

Löscht den ausgewählten Inhalt.

- \_WinCroSelPaste (3)

Ersetzt den ausgewählten Inhalt durch den Inhalt in der Zwischenablage.

- \_WinCroSelSelectAll (5) Wählt den

gesamten Inhalt aus.

Mögliche Laufzeitfehler

**ErrHdlInvalid**      Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

**ErrValueInvalid** In Argument (int1) wurde keine der gültigen Optionen angegeben.

### CodeEdit-Befehle

Befehle zum Verwenden eines CodeEdit-Objektes

Verwandte  
Befehle,

Siehe CodeEdit,  
Befehlsgruppen,  
Befehlsliste

### Befehle

- WinEditorBookmarkGet
- WinEditorBookmarkToggle
- WinEditorGetSelectedText
- WinEditorGetSelection
- WinEditorGoTo
- WinEditorHighlight
- WinEditorKeywordsAdd
- WinEditorKeywordsRemove
- WinEditorKeywordsUpdate
- WinEditorLoad
- WinEditorReplaceSelectedText
- WinEditorSave
- WinEditorSearch
- WinEditorSetSelection

## Kontakt

obj -> WinEditorBookmarkGet(int1[, var alpha2[, int3]])  
: logic



Lesezeichen im CodeEdit-Objekt setzen und entfernen

obj                      Deskriptor des

CodeEdit-Objektes

int1                    Zeile

var alpha2            Beschreibung (optional)

int4                   View-Nummer (optional)

Resultat            logic Lesezeichen gesetzt?



Verwandte Befehle,

Siehe                WinEditorBookmarkToggle()

Mit dieser Funktion kann im CodeEdit-Objekt (obj) ermittelt werden, ob in Zeile (int1) ein Lesezeichen gesetzt ist.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Optional kann die Beschreibung (alpha2) des Lesezeichens ermittelt werden.

In (int3) kann optional die Nummer des Views angegeben werden, in dem das Lesezeichen ermittelt werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

### Resultat

Als Resultat wird zurückgegeben, ob in der Zeile (int1) ein Lesezeichen gesetzt ist.

### Beispiel:

```
// Lesezeichen ermitteltSet # $CodeEdit->WinEditorBookmarkGet(42); // Lesezeichen und Beschreibung
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid      Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid   Ungültige oder nicht existierende View-Nummer (int3) angegeben.

obj -> WinEditorBookmarkToggle(int1[, alpha2[, int3[,  
logic4]]]) : logic



Lesezeichen im CodeEdit-Objekt setzen und entfernen

obj            Deskriptor des  
                 CodeEdit-Objektes

int1           Zeile

alpha2       Beschreibung (optional)

int4           View-Nummer (optional)

logic4       Explizit setzen / entfernen

                 (optional)

Resultat logic Lesezeichen gesetzt? —

Verwandte Befehle,

Siehe

WinEditorBookmarkGet()

Mit dieser Funktion können Lesezeichen im CodeEdit-Objekt (obj) gesetzt oder entfernt in der Zeile (int1) werden. Ist noch kein Lesezeichen gesetzt, wird es in der Zeile gesetzt, ansonsten das bestehende Lesezeichen entfernt.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Optional kann eine Beschreibung (alpha2) für das Lesezeichen angegeben werden.

In (int3) kann optional die Nummer des Views angegeben werden, in dem das Lesezeichen gesetzt bzw. entfernt werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.



Die gesetzten Lesezeichen werden für alle Views übernommen, bei denen die Eigenschaften FileName und EditorTextType auf den gleichen Wert gesetzt sind.

Ist Argument (logic4) angegeben, wird in der angegebenen Zeile (int1) bei true explizit ein Lesezeichen gesetzt, auch wenn bereits ein Lesezeichen vorhanden ist. Eine eventuell vorhandene Beschreibung wird hierbei durch (alpha2) überschrieben. Ist hier false oder NULL angegeben, wird ein vorhandenes Lesezeichen entfernt, jedoch nicht hinzugefügt, wenn keines existiert.

### Resultat

Als Resultat wird zurückgegeben, ob in der Zeile (int1) ein Lesezeichen gesetzt ist.

### Beispiel:

```
// Lesezeichen switchentSet # $CodeEdit->WinEditorBookmarkToggle(42) // Lesezeichen explizit setz
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid      Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid Ungültige oder nicht existierende View-Nummer (int3) angegeben.



## Kontakt

**: alpha**



obj	Deskriptor des CodeEdit-Objektes
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

**int2**      **View-Nummer (optional)**

Siehe WinEditorGetSelection(),  
WinEditorReplaceSelectedText()

 Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.


### Beispiel:

### Mögliche Laufzeitfehler:

**ErrHdlInvalid**

**ErrValueInvalid** Ungültige oder nicht existierende View-Nummer (int2) angegeben.

## Kontakt

obj -> WinEditorGetSelection(int1[, int2]) : point 

Selektionsbereich im CodeEdit-Objekt ermitteln

obj Deskriptor des CodeEdit-Objektes

Optionen

WinEditorSelGetCaretPos

Cursorposition

ermitteln

int1 WinEditorSelGetAnchorPos Ankerposition

ermitteln

WinEditorSelGetRange

Selektionslänge

ermitteln

int2 View-Nummer (optional)

Resultat point Ermittelte Koordinate / Länge



Verwandte Befehle,

WinEditorSetSelection(),

Siehe

WinEditorGetSelectedText(),

WinEditorReplaceSelectedText()

Mit dieser Funktion werden im CodeEdit-Objekt (obj) Informationen über den selektierten Bereich ermittelt.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Mit dem Argument (int1) wird der zu ermittelnde Wert angegeben. Folgende Konstanten können angegeben werden:

WinEditorSelGetCaretPos Cursorposition ermitteln

WinEditorSelGetAnchorPos Ankerposition ermitteln

WinEditorSelGetRange Selektionslänge ermitteln

In (int2) kann optional die Nummer des Views angegeben werden, in dem der Selektionsbereich ermittelt werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

### Resultat

Als Resultat wird eine Koordinate im Text zurückgegeben. Bei den Positionsangaben bestimmt die :x-Koordinate die Zeile und die :y-Koordinate die Spalte. Das erste Zeichen hat die Koordinate 1,1. Bei der Option WinEditorSelGetRange bestimmt die :x-Koordinate die Länge des selektierten Textes. Die :y-Koordinate ist 0.

Beispiel:

```
// Selektionsanfang und -Ende ermittelntCaret # $CodeEdit->WinEditorGetSelection( _WinEditorSelGet
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid Ungültige oder nicht existierende View-Nummer (int2) angegeben.

## Kontakt

obj -> WinEditorGoTo(point1[, logic2[,  
int3]])



Im CodeEdit-Objekt zur Position scrollen  
Deskriptor des

obj

CodeEdit-Objektes

point1 Position

logic2 Nur scrollen (optional)

View-Nummer

int3

(optional)

Verwandte Befehle,

Siehe WinEditorSetSelection(),

WinEditorHighlight()

Mit dieser Funktion wird im CodeEdit-Objekt (obj) zur angegebenen Position gescrollt.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Die Position wird mit (point1) definiert.



Bei der Positionsangaben bestimmt die :x-Koordinate die Zeile und die :y-Koordinate die Spalte. Das erste Zeichen hat die Koordinate 1,1.

Wird im optionalen Argument (logic2) true (Default) angegeben, wird nur zu den Koordinaten gescrollt. Durch Angabe von false wird zusätzlich die Cursor-Position auf die angegebene Position gesetzt.

In (int3) kann optional die Nummer des Views angegeben werden, in dem selektiert werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( \_WinCount) ermittelt werden.

Beispiel:

```
// Zu Zeile 4, Spalte 2 scrollen$CodeEdit->WinEditorGoTo(PointMake(2, 4)); // Cursorposition in Z
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid Ungültige oder nicht existierende View-Nummer (int3) angegeben.

## Kontakt

obj -> WinEditorHighlight(point1[,  
logic2[, point3[, int4]]]) : rect



Text im CodeEdit-Objekt hervorheben  
Deskriptor des

obj

CodeEdit-Objektes

point1 Startposition

Hervorhebung löschen

logic2 (optional)

point3 Endposition (optional)

int4 View-Nummer (optional)

Siehe

WinEditorSetSelection()

Mit dieser Funktion kann ein Bereich im CodeEdit-Objekt (obj) hervorgehoben werden. Die Hervorhebung findet in der Farbe ThemeColEditorCustomHighlight statt.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Die Startposition wird mit (point1) definiert.

Die Endposition wird mit (point3) angegeben. Wird (point3) nicht angegeben, geht die Hervorhebung bis zum Ende des Textes.



Bei den Positionsangaben bestimmt die :x-Koordinate die Zeile und die :y-Koordinate die Spalte. Das erste Zeichen hat die Koordinate 1,1. Negative :x-Koordinaten werden auf das Ende des Textes gesetzt.

Parameter (logic2) definiert, ob eine bisher vorhandene Hervorhebung entfernt werden soll.

Wird der Parameter nicht angegeben, wird er auf true gesetzt.

In (int4) kann optional die Nummer des Views angegeben werden, in dem der Selektionsbereich ermittelt werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.



Die Hervorhebung wird für alle Views übernommen, bei denen die Eigenschaften FileName und EditorTextType auf den gleichen Wert gesetzt sind.

Beispiel:

```
// Bereich hervorheben$CodeEdit->WinEditorHighlight(PointMake(5, 5), false, PointMake(5, 10));//
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid Ungültige oder nicht existierende View-Nummer (int2) angegeben.



**WinEditorKeywordsAdd(alpha1, int2) : int**  
 Selbstdefinierte Schlüsselwörter zu den  
CodeEdit-Objekten

hinzufügen

**alpha1** Schlüsselwörter mit '|' (Pipe) getrennt  
 Typ der Schlüsselwörter  
WinEditorKeyDatatype Datentyp  
WinEditorKeyKeyword Schlüsselwort  
**int2**  
WinEditorKeyFunction Funktion  
WinEditorKeyConstant Konstante  
WinEditorKeyCustom kundenspezifisch  
 Fehlercode

**Resultat** int ErrOk Kein Fehler aufgetreten





ErrUnavailable Kein CodeEdit-Objekt erzeugt  
Verwandte Befehle, WinEditorKeywordsRemove(),

Siehe

WinEditorKeywordsUpdate()

Mit dieser Funktion können eigene Schlüsselwörter zu allen CodeEdit-Objekten hinzugefügt werden. Damit diese im CodeEdit-Objekt hervorgehoben werden, muss die Eigenschaft EditorTextType auf WinEditorTextTypePrc gesetzt sein. Hinzugefügte

Schlüsselwörter werden zusätzlich in die Auto-Vervollständigungsliste (  +  ) aufgenommen.



Diese Methode kann frühestens im EvtCreated des Elternfensters des ersten CodeEdit-Objektes verwendet werden.



Die Liste der Schlüsselwörter wird automatisch zurückgesetzt, wenn die letzte laufende Prozedur verlassen wird (Prozedur nach Start oder Testprozedur im Standard-Client bzw. Rückkehr in den Designer).



Die Schlüsselwörter werden nicht auf vorhandensein geprüft. Wird ein Schlüsselwort mehrfach hinzugefügt, erscheint es mehrfach in der Auto-Vervollständigungsliste.

Im Argument (alpha1) müssen ein oder mehrere hinzuzufügende Schlüsselwörter angegeben werden. Mehrere Schlüsselwörter werden durch '|' (Pipe) getrennt.



Das erste Zeichen der Schlüsselwörter muss ein Unterstrich oder ein Buchstabe sein. Ab dem zweiten Zeichen dürfen zusätzlich Punkte und Ziffern verwendet werden.

In (int2) wird der Typ der Schlüsselwörter angegeben. Der entsprechende Typ definiert die Hervorhebung im CodeEdit-Objekt. Folgende Typen können angegeben werden:

Konstante	Typ	Stil
<u>WinEditorKeyDatatype</u>	Datentyp	Fett mit Farbe <u>ThemeColEditorDataTypes</u>
<u>WinEditorKeyKeyword</u>	Schlüsselwort	Fett mit Farbe <u>ThemeColEditorKeywords</u>
<u>WinEditorKeyFunction</u>	Funktion	Farbe <u>ThemeColEditorFunctions</u>
<u>WinEditorKeyConstant</u>	Konstante	Kursiv mit Farbe <u>ThemeColEditorConstants</u>
<u>WinEditorKeyCustom</u>	kundenspezifisch	Fett mit Farbe <u>ThemeColEditorCKeywords</u>

**Resultat**

## Kontakt

Das Resultat ist **ErrOk**, wenn das Hinzufügen der Schlüsselwörter erfolgreich war. Ist noch kein **CodeEdit**-Objekt fertig initialisiert, wird **ErrUnavailable** zurückgegeben.



Die Liste der Schlüsselwörter muss für jedes **CodeEdit**-Objekt separat mittels **WinEditorKeywordsUpdate()** aktualisiert werden. Diese Notwendigkeit entfällt, wenn die Schlüsselwörter hinzugefügt wurden, bevor eine Datei im **CodeEdit** geöffnet wird.

Beispiel:

```
// Schlüsselwörter hinzufügenErr # WinEditorKeywordsAdd('MyFunction1|MyFunction2', _WinEditorKey
```

Mögliche Laufzeitfehler:

Leeres Schlüsselwort oder ungültiges Zeichen im Schlüsselwort  
**ErrValueInvalid** (alpha1) oder ungültiger Typ (int2) angegeben.

## Kontakt

**WinEditorKeywordsRemove([int1[, alpha2]]) : int** 

Schlüsselwörter der CodeEdit-Objekte entfernen

int1      Modus (optional)

alpha2    Zu entfernende Schlüsselwörter

Fehlercode

Resultat int ErrOk      Kein Fehler aufgetreten 

ErrUnavailable Kein CodeEdit-Objekt erzeugt

Verwandte Befehle, WinEditorKeywordsAdd(),

Siehe

WinEditorKeywordsUpdate()

Mit dieser Funktion werden die Schlüsselwörter aller CodeEdit-Objekte zurückgesetzt. Damit werden alle mit WinEditorKeywordsAdd() hinzugefügten Wörter nicht mehr hervorgehoben.



Diese Methode kann frühestens im EvtCreated des Elternfensters des ersten CodeEdit-Objektes verwendet werden.

Mit dem optionalen Argument (int1) kann definiert werden, welche Schlüsselwörter entfernt werden. Folgende Konstanten können angegeben werden:

WinEditorKeyRemoveCustom Entfernt alle Schlüsselwörter, die mit WinEditorKeywordsAdd() hinzugefügt wurden

(Default)

WinEditorKeyRemoveAll Entfernt alle Schlüsselwörter, inklusive der

CONZEPT 16 Schlüsselwörter

WinEditorKeyRemoveList Entfernt die in (alpha2)angegebenen Schlüsselwörter

In dem optionalen Argument (alpha2) werden die zu entfernenden Schlüsselwörter bei dem Modus WinEditorKeyRemoveList angegeben. Mehrere Schlüsselwörter werden durch '|' (Pipe) getrennt.

### Resultat

Das Resultat ist ErrOk, wenn das Löschen der Schlüsselwörter erfolgreich war. Ist noch kein CodeEdit-Objekt fertig initialisiert, wird ErrUnavailable zurückgegeben.



Die Liste der Schlüsselwörter muss für jedes CodeEdit-Objekt separat mittels WinEditorKeywordsUpdate() aktualisiert werden. Diese Notwendigkeit entfällt, wenn die Schlüsselwörter zurückgesetzt wurden, bevor eine Datei im CodeEdit geöffnet wird.

### Beispiel:

```
// Selbst hinzugefügte Schlüsselwörter entfernenErr # WinEditorKeywordsRemove();// CodeEdit aktu
```

## Kontakt

obj -> WinEditorKeywordsUpdate([int1]) : int 

Schlüsselwörter im CodeEdit-Objekt aktualisieren

obj            Deskriptor des CodeEdit-Objektes

int4           View-Nummer (optional)

Fehlercode

ErrOk

Kein Fehler aufgetreten

Resultat logic

ErrUnavailable EditorTextType !=



WinEditorTextTypePrc

Verwandte Befehle, WinEditorKeywordsAdd(),

Siehe

WinEditorKeywordsRemove()

Mit dieser Funktion werden die mittels WinEditorKeywordsAdd() hinzugefügten bzw. mit WinEditorKeywordsRemove() entfernten Schlüsselwörter im CodeEdit-Objekt (obj) angewendet. Die Eigenschaft EditorTextType muss hierzu auf WinEditorTextTypePrc gesetzt sein.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

In (int1) kann optional die Nummer des Views angegeben werden, in dem die Schlüsselwörter aktualisiert werden sollen. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) bewirkt eine Aktualisierung in allen Views. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

### Resultat

Das Resultat ist ErrOk, wenn die Schlüsselwörter erfolgreich aktualisiert wurden. Ist die Eigenschaft EditorTextType nicht auf WinEditorTextTypePrc gesetzt, wird ErrUnavailable zurückgegeben.

### Beispiel:


```
// Schlüsselwörter hinzufügenWinEditorKeywordsAdd('MyFunction1|MyFunction2', _WinEditorKeyFunctio
```

### Mögliche Laufzeitfehler:

Der Deskriptor des CodeEdit (obj) ist ungültig oder das View mit der ErrHdlInvalid Nummer (int1) ist nicht vorhanden.




## Kontakt

obj -> WinEditorLoad([alpha1[,  int2[, int3[, int4]]]]) : int

Text im CodeEdit-Objekt laden  
obj Deskriptor des CodeEdit-Objektes

alpha1 Name der Prozedur / des Textes  
(optional)  
Quelle (optional)  
WinStreamNameText Interne(n)  
Prozedur / Text

int2 laden  
WinStreamNameFile Externen Text  
laden  
Typ (optional)  
WinEditorTextTypeAuto Texttyp anhand  
der Dateiendung  
ermitteln  
WinEditorTextTypePrc Text als  
Prozedur laden  
WinEditorTextTypeText Text als  
normalen Text  
laden  
WinEditorTextTypeXml Text als  
XML-Dokument  
laden  
WinEditorTextTypeJson Text als  
JSON-Dokument  
laden

int4 View-Nummer (optional)  
Fehlercode  
rOk Speichern erfolgreich  
rNoRec Prozedur / Text  
(alpha1)nicht  
vorhanden 

Resultat int  
rNoRights Benutzerberechtigung  
nicht ausreichend  
ErrFsi... Fehler für externe  
Dateioperationen

Siehe Verwandte Befehle, WinEditorSave()

Mit dieser Funktion wird der Inhalt einer Prozedur / eines Textes in das CodeEdit-Objekt (obj) geladen.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Optional kann in (alpha1) der Name angegeben werden. Ist der Text bisher noch nicht geladen, muss hier ein Name definiert werden.

Als Quelle (int2) können folgende Konstanten angegeben werden:

## Kontakt

WinStreamNameText Interne(n) Prozedur / Text laden

WinStreamNameFile Externen Text laden

Wird das Argument (int2) nicht angegeben, wird die Quelle anhand des Namens (alpha1) bzw. der Eigenschaft FileName ermittelt.

Folgende Typen (int3) können angegeben werden:

WinEditorTextTypeAuto Texttyp anhand der Dateiendung ermitteln

WinEditorTextTypePrc Text als Prozedur laden

WinEditorTextTypeText Text als normalen Text laden

WinEditorTextTypeXml Text als XML-Dokument laden

WinEditorTextTypeJson Text als JSON-Dokument laden

In (int4) kann optional die Nummer des Views angegeben werden, in dem der Text geladen werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. Wird als View-Nummer 0 (oder nicht angegeben) angegeben, werden alle dargestellten Texte geladen. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

### Resultat

Das Resultat ist ErrOk, wenn alle Texte geladen werden konnten. Neben den Fehlerwerten für externe Dateioperationen kann für interne Dokumente einer der folgenden Fehlerwerte zurückgegeben werden:

rNoRec Prozedur / Text (alpha1) nicht vorhanden rNoRights

Benutzerberechtigung nicht ausreichend. Beispiel:

```
// Geladenes Dokument mit gleichen Einstellungen erneut ladentResult # $CodeEdit->WinEditorLoad()
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid Ungültige Quelle (int2), ungültiger Typ (int3) oder ungültige bzw. nicht existierende View-Nummer (int4) angegeben.

obj -> WinEditorReplaceSelectedText(alpha1[,  
handle2[, int3]])



Selektierten Text im CodeEdit-Objekt ersetzen  
Deskriptor des

obj

CodeEdit-Objektes

alpha1 Ersetzungstext

handle2 Deskriptor eines

Memory-Objektes (optional)

int3 View-Nummer (optional)

Verwandte Befehle,

Siehe WinEditorGetSelection(),

WinEditorGetSelectedText()

Mit dieser Funktion wird im CodeEdit-Objekt (obj) der selektierte Text ersetzt. Wird ein Memory-Objekt (handle2) wird der selektierte Text den den Inhalt dieses ersetzt. Andernfalls wird der Ersetzungstext (alpha1) verwendet.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

In (int3) kann optional die Nummer des Views angegeben werden, in dem der selektierte Text ersetzt werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

Beispiel:

```
// Selektierten Text ersetzen$CodeEdit->WinEditorReplaceSelectedText('replaced content');
```

Mögliche Laufzeitfehler:

ErrHdlInvalid

Der Deskriptor des CodeEdit (obj) oder des Memory-Objektes  
(handle2) ist ungültig.

ErrValueInvalid Ungültige oder nicht existierende View-Nummer (int3) angegeben.

## Kontakt

obj -> WinEditorSave([alpha1[, int2[,  
int3[, logic4[, int5]]]]) : int Text im  
CodeEdit-Objekt speichern



obj Deskriptor des CodeEdit-Objektes

alpha1 Name der Prozedur / des Textes  
(optional)

Ziel (optional)

\_WinStreamNameText Interne(n)

Prozedur / Text

int2

speichern

\_WinStreamNameFile Externen Text

speichern

Typ (optional)

\_WinEditorTextTypeAuto Texttyp

beibehalten

\_WinEditorTextTypePrc

Text als

Prozedur

speichern

\_WinEditorTextTypeText Text als

normalen Text

speichern

\_WinEditorTextTypeXml

Text als

XML-Dokument

speichern

\_WinEditorTextTypeJson Text als

JSON-Dokument

speichern

Optionen (optional)

\_WinEditorSaveOverwrite Prozedur /

int4

Text

überschreiben

int5

View-Nummer (optional)

Fehlercode

\_rOk

Speichern erfolgreich

\_rLocked

Text / Prozedur

(alpha1) gesperrt

\_rExists

Text / Prozedur

(alpha1) existiert

Resultat int

bereits



\_rNoRights

Benutzerberechtigung

nicht ausreichend

\_rDeadlock

Verklemmung

aufgetreten

\_ErrFsi...

Fehler für externe

Dateioperationen

Siehe Verwandte Befehle, WinEditorLoad()

Mit dieser Funktion wird der Inhalt des CodeEdit-Objektes (obj) gespeichert.

## Kontakt



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Optional kann in (alpha1) der Name angegeben werden. Ist der Text bisher noch nicht gespeichert, muss hier ein Name definiert werden.

Soll ein bestehendes internes Dokument überschrieben werden, das zuvor nicht geladen wurde, muss die Option (int4) \_WinEditorSaveOverwrite angegeben werden.

Als Ziel (int2) können folgende Konstanten angegeben werden:

\_WinStreamNameText Interne(n) Prozedur / Text laden

\_WinStreamNameFile Externen Text laden

Wird das Argument (int2) nicht angegeben, wird das Ziel die Quelle aus WinEditorLoad() verwendet.

Folgende Typen (int3) können angegeben werden:

\_WinEditorTextTypeAuto Texttyp beibehalten

\_WinEditorTextTypePrc Text als Prozedur speichern

\_WinEditorTextTypeText Text als normalen Text speichern

\_WinEditorTextTypeXml Text als XML-Dokument speichern

\_WinEditorTextTypeJson Text als JSON-Dokument speichern

In (int5) kann optional die Nummer des Views angegeben werden, in dem der Text gespeichert werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. Wird als View-Nummer 0 (oder nicht angegeben) angegeben, werden alle dargestellten Texte gespeichert. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( \_WinCount) ermittelt werden.

## Resultat

Das Resultat ist \_ErrOk, wenn alle Texte gespeichert werden konnten. Neben den Fehlerwerten für externe Dateioperationen kann für interne Dokumente einer der folgenden Fehlerwerte zurückgegeben werden:

\_rLocked Text / Prozedur (alpha1) gesperrt.

\_rExists Text / Prozedur (alpha1) existiert bereits. Option (int4)

\_WinEditorSaveOverwrite muss zum Überschreiben angegeben werden.

\_rNoRights Benutzerberechtigung nicht ausreichend.

\_rDeadlock Verklemmung aufgetreten.

Beispiel:

```
// Geladenes Dokument mit gleichen Einstellungen speicherntResult # $CodeEdit->WinEditorSave() ;//
```

Mögliche Laufzeitfehler:

\_ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

\_ErrValueInvalid Ungültiges Ziel (int2), ungültiger Typ (int3), ungültige Optionen (int4) oder ungültige bzw. nicht existierende View-Nummer (int5)

**angegeben.**

## Kontakt

obj -> WinEditorSearch(alpha1[, int2[, point3[,  
alpha4[, int5]]]]) : rect



Text im CodeEdit-Objekt suchen und ersetzen

obj Deskriptor des CodeEdit-Objektes

alpha1 Suchbegriff

Optionen (optional)

\_WinEditorSearchRev

Rückwärts suchen

\_WinEditorSearchInSel

Im selektierten

\_WinEditorSearchGoTo

Bereich suchen

Zu Suchtreffer

\_WinEditorSearchRegExp

springen

Regulären Ausdruck

\_WinEditorSearchWholeWord

suchen

Ganzes Wort suchen

int2 \_WinEditorSearchCI

Groß-/Kleinschreibung

\_WinEditorSearchHighlightWord

ignorieren

Aktuellen Suchtreffer

\_WinEditorSearchHighlightOccurrences Andere Suchtreffer

hervorheben

\_WinEditorSearchReplace

hervorheben

Aktuellen Suchtreffer

\_WinEditorSearchReplaceAll

ersetzen

Alle Suchtreffer

ersetzen

point3 Startposition (optional)

alpha4 Ersetzungsbegriff (optional)

int5 View-Nummer (optional)

Resultat rect Position des Suchtreffers



Siehe Verwandte Befehle, WinEditorGetSelection()

Mit dieser Funktion wird der Inhalt des CodeEdit-Objektes (obj) nach der Zeichenfolge (alpha1) durchsucht.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Als Suchoption (int2) können optional folgende Konstanten angegeben werden:

\_WinEditorSearchRev

Rückwärts suchen

\_WinEditorSearchInSel

Im selektierten Bereich suchen

\_WinEditorSearchGoTo

Zu Suchtreffer springen

\_WinEditorSearchRegExp

Regulären Ausdruck suchen

\_WinEditorSearchWholeWord

Ganzes Wort suchen

\_WinEditorSearchCI

Groß-/Kleinschreibung ignorieren

\_WinEditorSearchHighlightWord

Aktuellen Suchtreffer hervorheben

\_WinEditorSearchHighlightOccurrences Andere Suchtreffer hervorheben

\_WinEditorSearchReplace

Aktuellen Suchtreffer ersetzen

## Kontakt

### WinEditorSearchReplaceAll

Alle Suchtreffer ersetzen

Die Optionen können miteinander kombiniert werden.

Mit den Optionen WinEditorSearchReplace und WinEditorSearchReplaceAll werden die Suchtreffer durch den Ersetzungsbegriff (alpha4) ersetzt. Ist dieser leer, werden die Suchtreffer entfernt.

Die Startposition kann optional mit (point3) definiert werden.



Bei den Positionsangaben bestimmt die :x-Koordinate die Zeile und die :y-Koordinate die Spalte. Das erste Zeichen hat die Koordinate 1,1. Wird der Punkt -1,-1 angegeben, wird die Positionsangabe ignoriert. Abhängig davon, ob die Suchoption WinEditorSearchRev angegeben ist, wird vom Anfang oder vom Ende des Textes gesucht.

In (int5) kann optional die Nummer des Views angegeben werden, in dem gesucht werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

### Resultat

Als Resultat wird die Position des Suchtreffers zurückgegeben. Hierbei bestimmt die :left-Koordinate die Anfangszeile, die :top-Koordinate die Anfangsspalte, die :right-Koordinate die Endzeile und die :bottom-Koordinate die Endspalte des Treffers.

### Beispiel:

```
// Wort suchen ohne MarkierungRect # $CodeEdit->WinEditorSearch('Hallo');// Wort ab bestimmter P
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid Ungültige Option (int2) oder ungültige oder nicht existierende View-Nummer (int5) angegeben.



## Kontakt



obj -> WinEditorSetSelection(point1[, point2[, int3[, int4]])

Selektionsbereich im CodeEdit-Objekt setzen oder hinzufügen obj

Deskriptor des CodeEdit-Objektes

point1 Startpunkt / Cursor-Position

point2 Endpunkt (optional)

Optionen (optional)

WinEditorSelSetDefault Normale Selektion

int3

WinEditorSelSetBlock setzen (Default)  
Blockselektion

WinEditorSelSetMulti setzen  
Normalen  
Selektionsbereich  
hinzufügen

int4 View-Nummer (optional)

Verwandte Befehle,

Siehe WinEditorGetSelection(),  
WinEditorHighlight(), WinEditorGoTo()

Mit dieser Funktion wird im CodeEdit-Objekt (obj) ein Bereich selektiert.



Diese Methode kann frühestens im EvtCreated des Elternfensters verwendet werden.

Die Startposition wird mit (point1) definiert.

Die Endposition wird mit (point2) angegeben. Wird (point2) nicht angegeben, wird nur der Cursor an die Startposition (point1) gesetzt.



Bei den Positionsangaben bestimmt die :x-Koordinate die Zeile und die :y-Koordinate die Spalte. Das erste Zeichen hat die Koordinate 1,1. Negative :x-Koordinaten werden auf das Ende des Textes gesetzt.

Mit dem Argument (int3) kann optional der Typ der Markierung definiert werden.

Folgende Konstanten können angegeben werden:

WinEditorSelSetDefault Normale Selektion setzen (Default)

WinEditorSelSetBlock Blockselektion setzen

WinEditorSelSetMulti Normalen Selektionsbereich hinzufügen

In (int4) kann optional die Nummer des Views angegeben werden, in dem selektiert werden soll. Die Views können mit den Nummern 1 bis 4 angesprochen werden. View-Nummer 0 (oder nicht angegeben) ist gleichbedeutend mit 1. Die Anzahl der Views kann mit \$CodeEdit->WinInfo( WinCount) ermittelt werden.

Beispiel:

```
// Kompletten Text markieren$CodeEdit->WinEditorSetSelection(PointMake(1, 1), PointMake(-1, 0));/
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor des CodeEdit (obj) ist ungültig.

ErrValueInvalid

## **Kontakt**

**Ungültige Option (int3) oder ungültige oder nicht existierende View-Nummer (int4) angegeben.**

Konstanten für CodeEdit-Befehle

Konstanten für CodeEdit-Befehle

Siehe CodeEdit-Befehle

- WinEditorKeyConstant
- WinEditorKeyCustom
- WinEditorKeyDatatype
- WinEditorKeyFunction
- WinEditorKeyKeyword
- WinEditorKeyRemoveAll
- WinEditorKeyRemoveCustom
- WinEditorKeyRemoveList
- WinEditorSaveOverwrite
- WinEditorSearchCI
- WinEditorSearchGoTo
- WinEditorSearchHighlightOccurrences
- WinEditorSearchHighlightWord
- WinEditorSearchInSel
- WinEditorSearchRegExp
- WinEditorSearchReplace
- WinEditorSearchReplaceAll
- WinEditorSearchRev
- WinEditorSearchWholeWord
- WinEditorSelGetAnchorPos
- WinEditorSelGetCaretPos
- WinEditorSelGetRange
- WinEditorSelSetBlock
- WinEditorSelSetDefault
- WinEditorSelSetMulti
- WinEditorTextTypeAuto
- WinEditorTextTypeJson
- WinEditorTextTypePrc
- WinEditorTextTypeText
- WinEditorTextTypeXml

**\_WinEditorKeyConstant**

Schlüsselwort ist eine Konstante

Wert 3

**Siehe** Verwandte Befehle,

WinEditorKeywordsAdd()

Option bei WinEditorKeywordsAdd(), durch die die hinzugefügten Schlüsselwörter als Konstante (Kursiv mit Farbe ThemeColEditorConstants) dargestellt werden.

## Kontakt

`_WinEditorKeyCustom`

Schlüsselwort ist kundenspezifisch

Wert 5

**Siehe** Verwandte Befehle,

`WinEditorKeywordsAdd()`

Option bei `WinEditorKeywordsAdd()`, durch die die hinzugefügten Schlüsselwörter als kundenspezifisches Schlüsselwort (Fett mit Farbe `ThemeColEditorCKeywords`) dargestellt werden.

**\_WinEditorKeyDatatype**

Schlüsselwort ist ein Datentyp

Wert 0

**Siehe** Verwandte Befehle,

WinEditorKeywordsAdd()

Option bei WinEditorKeywordsAdd(), durch die die hinzugefügten Schlüsselwörter als Datentyp (Fett mit Farbe ThemeColEditorDataTypes) dargestellt werden.

**\_WinEditorKeyFunction**

Schlüsselwort ist eine Funktion

Wert 2

**Siehe** Verwandte Befehle,

WinEditorKeywordsAdd()

Option bei WinEditorKeywordsAdd(), durch die die hinzugefügten Schlüsselwörter als Funktion (Farbe ThemeColEditorFunctions) dargestellt werden.

**\_WinEditorKeyKeyword**

Schlüsselwort ist ein Schlüsselwort

Wert 1

**Siehe** Verwandte Befehle,

WinEditorKeywordsAdd()

Option bei WinEditorKeywordsAdd(), durch die die hinzugefügten Schlüsselwörter als Schlüsselwort (Fett mit Farbe ThemeColEditorKeywords) dargestellt werden.



## Kontakt

**\_WinEditorKeyRemoveAll**

Alle Schlüsselwörter entfernen

Wert 1

Siehe **WinEditorKeywordsRemove()**

Option bei **WinEditorKeywordsRemove()**, durch die alle Schlüsselwörter entfernt werden können.

## Kontakt

**\_WinEditorKeyRemoveCustom**

Selbst hinzugefügte Schlüsselwörter entfernen Wert 0

Siehe **WinEditorKeywordsRemove()**

Option bei **WinEditorKeywordsRemove()**, durch die alle mit **WinEditorKeywordsAdd()** hinzugefügten Schlüsselwörter entfernt werden können.

## Kontakt

**\_WinEditorKeyRemoveList**

Ausgewählte Schlüsselwörter entfernen

Wert 2

Siehe **WinEditorKeywordsRemove()**

Option bei **WinEditorKeywordsRemove()**, durch die ausgewählte Schlüsselwörter entfernt werden können.

**\_WinEditorSaveOverwrite**

**Internes Dokument überschreiben**

**Wert 1 / 0x0001**

**Verwandte**

**Siehe Befehle,**

**WinEditorSave()**

**Option bei WinEditorSave() durch die ein bestehendes Dokument, welches nicht aktuell im CodeEdit-Objekt dargestellt wird, überschrieben werden kann.**

`_WinEditorSearchCI`

Groß-/Kleinschreibung ignorieren

Wert 32 / 0x0020

Siehe Verwandte Befehle,

Option bei WinEditorSearch() durch die die Groß-/Kleinschreibung beim Suchen ignoriert werden kann.

**\_WinEditorSearchGoTo**

**In Trefferzeile springen**

**Wert 4 / 0x0004**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die in die Zeile des Suchtreffers gesprungen wird.**

**\_WinEditorSearchHighlightOccurences**

Weitere Suchtreffer hervorheben

Wert 128 / 0x0080

Siehe Verwandte Befehle,

Option bei WinEditorSearch() durch die alle weiteren außer dem aktuellen Suchtreffer hervorgehoben werden.

**\_WinEditorSearchHighlightWord**

**Aktuellen Suchtreffer hervorheben**

**Wert 64 / 0x0040**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die der aktuelle Suchtreffer hervorgehoben wird.**



**\_WinEditorSearchInSel**

**In selektiertem Bereich suchen**

**Wert 2 / 0x0002**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die nur im selektierten Bereich gesucht wird.**

## Kontakt

**\_WinEditorSearchRegExp**

**Regulären Ausdruck verwenden**

**Wert 8 / 0x0008**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die der Suchbegriff als regulärer Ausdruck interpretiert wird.**

## Kontakt

**\_WinEditorSearchReplace**

**Aktuellen Suchtreffer ersetzen**

**Wert 256 / 0x0100**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die der aktuelle Suchtreffer ersetzt wird.**

**`_WinEditorSearchReplaceAll`**

Alle Suchtreffer ersetzen

Wert 512 / 0x0200

Siehe Verwandte Befehle,

Option bei WinEditorSearch() durch die alle Suchtreffer ersetzt werden.

**\_WinEditorSearchRev**

**Rückwärts suchen**

**Wert 1 / 0x0001**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die rückwärts gesucht wird.**

**\_WinEditorSearchWholeWord**

**Nach ganzen Wörtern suchen**

**Wert 16 / 0x0010**

**Siehe Verwandte Befehle,**

**Option bei WinEditorSearch() durch die die Suchergebnisse auf ganze Wörter beschränkt werden kann.**

**Der Suchbegriff wird nur dann gefunden, wenn vor und nach ihm ein Worttrennzeichen steht. Worttrennzeichen sind alle Zeichen mit Ausnahme von Buchstaben oder Zahlen. Das Zeilenende trennt ebenfalls Wörter voneinander.**

**\_WinEditorSelGetAnchorPos**

Ankerposition aus Selektion ermitteln

Wert 1

**WinEditorGetSelection()**,

Siehe **\_WinEditorSelGetCaretPos**,

**WinEditorSelGetRange**

Option bei **WinEditorGetSelection()**, mit der die Ankerposition der aktuellen Selektion im **CodeEdit**-Objekt ermittelt wird.

**\_WinEditorSelGetCaretPos**

Cursorposition aus Selektion ermitteln  
Wert 0

WinEditorGetSelection(),  
Siehe \_WinEditorSelGetAnchorPos,  
WinEditorSelGetRange

Option bei WinEditorGetSelection(), mit der die Cursorposition der aktuellen Selektion im CodeEdit-Objekt ermittelt wird.



**\_WinEditorSelGetRange**

Länge der Selektion ermitteln

Wert 2

**WinEditorGetSelection()**,

Siehe **\_WinEditorSelGetCaretPos**,

**WinEditorSelGetAnchorPos**

Option bei **WinEditorGetSelection()**, mit der die Länge der aktuellen Selektion im **CodeEdit**-Objekt ermittelt wird.

**\_WinEditorSelSetBlock**

Text mit Blockseletion selektieren

Wert 1

**WinEditorSetSelection()**,

Siehe **\_WinEditorSelSetDefault**,

**WinEditorSelSetMulti**

Option bei **WinEditorSetSelection()**, mit der eine Blockselektion gesetzt wird.

**\_WinEditorSelSetDefault**

Text selektieren

Wert 0

**WinEditorSetSelection()**,

Siehe **\_WinEditorSelSetBlock**,

**WinEditorSelSetMulti**

Option bei **WinEditorSetSelection()**, mit der eine normale Selektion gesetzt wird.

**\_WinEditorSelSetMulti**

Selektion hinzufügen

Wert 2

**WinEditorSetSelection()**,

Siehe **WinEditorSelSetDefault**,

**WinEditorSelSetBlock**

Option bei **WinEditorSetSelection()**, mit die bestehenden Selektionen erhalten bleiben und ein weiterer Bereich selektiert wird.

**\_WinEditorTextTypeAuto**

Texttyp ermitteln oder beibehalten

Wert -1

Siehe WinEditorLoad(),

Diese Option kann bei den Befehlen WinEditorLoad() und WinEditorSave() angegeben werden, um den Typ des zu ladenden Dokumentes automatisch zu ermitteln bzw. zu speichernden Dokumentes beizubehalten. Beim Laden wird automatisch die Eigenschaft EditorTextType auf den ermittelten Typ gesetzt.

**\_WinEditorTextTypeJson**

Darzustellender Text ist ein JSON-Dokument Wert 3

**EditorTextType,**

Siehe **WinEditorLoad(),**

**WinEditorSave()**

Wird die Eigenschaft **EditorTextType** auf den Wert **\_WinEditorTextTypeJson** gesetzt, wird im **CodeEdit**-Objekt ein JSON-Dokument angezeigt.

Diese Option kann zusätzlich bei den Befehlen **WinEditorLoad()** und **WinEditorSave()** angegeben werden, um den Typ des zu ladenden bzw. zu speichernden Dokumentes festzulegen. Hierbei wird automatisch die Eigenschaft **EditorTextType** gesetzt.

**\_WinEditorTextTypePrc**

Darzustellender Text ist eine Prozedur

Wert 0

**EditorTextType,**

Siehe **WinEditorLoad(),**

**WinEditorSave()**

Wird die Eigenschaft **EditorTextType** auf den Wert **\_WinEditorTextTypePrc** gesetzt, wird im **CodeEdit**-Objekt eine Prozedur angezeigt.

Diese Option kann zusätzlich bei den Befehlen **WinEditorLoad()** und **WinEditorSave()** angegeben werden, um den Typ des zu ladenden bzw. zu speichernden Dokumentes festzulegen. Hierbei wird automatisch die Eigenschaft **EditorTextType** gesetzt.

**\_WinEditorTextTypeText**

Darzustellender Text ist ein Text

Wert 1

**EditorTextType**,

Siehe **WinEditorLoad()**,

**WinEditorSave()**

Wird die Eigenschaft **EditorTextType** auf den Wert **\_WinEditorTextTypeText** gesetzt, wird im **CodeEdit**-Objekt ein Text angezeigt.

Diese Option kann zusätzlich bei den Befehlen **WinEditorLoad()** und **WinEditorSave()** angegeben werden, um den Typ des zu ladenden bzw. zu speichernden Dokumentes festzulegen. Hierbei wird automatisch die Eigenschaft **EditorTextType** gesetzt.



**\_WinEditorTextTypeXml**

Darzustellender Text ist ein XML-Dokument Wert

2

**EditorTextType**,

Siehe **WinEditorLoad()**,

**WinEditorSave()**

Wird die Eigenschaft **EditorTextType** auf den Wert **\_WinEditorTextTypeXml** gesetzt, wird im **CodeEdit**-Objekt ein XML-Dokument angezeigt.

Diese Option kann zusätzlich bei den Befehlen **WinEditorLoad()** und **WinEditorSave()** angegeben werden, um den Typ des zu ladenden bzw. zu speichernden Dokumentes festzulegen. Hierbei wird automatisch die Eigenschaft **EditorTextType** gesetzt.

## CtxDocEdit-Befehle

Befehle und Konstanten für CtxDocEdit-Objekt

Verwandte  
Befehle,

Siehe CtxDocEdit,  
Befehlsgruppen,  
Befehlsliste

### Befehle

- WinDocLoadBin
- WinDocLoadName
- WinDocPrint
- WinDocSaveBin
- WinDocSaveName
- WinDocUserDictAddName
- WinDocUserDictRemoveName

### Konstanten

- WinDocLoadAscii
- WinDocLoadAuto
- WinDocLoadDoc
- WinDocLoadDocX
- WinDocLoadHtml
- WinDocLoadInsert
- WinDocLoadMix
- WinDocLoadOEM
- WinDocLoadRtf
- WinDocSaveAscii
- WinDocSaveAuto
- WinDocSaveDoc
- WinDocSaveDocX
- WinDocSaveHtml
- WinDocSaveMark
- WinDocSaveMix
- WinDocSaveOEM
- WinDocSavePdf
- WinDocSaveRtf
- WinStreamNameFile
- WinStreamNameText

obj -> WinDocLoadBin(handle1, int2[,  
alpha3]) : int



Binäres Objekt in CtxDocEdit-Objekt laden

obj Objekt (CtxDocEdit-Objekt)

handle1 Deskriptor des binären Objekts

	Modus	
	<u>WinDocLoadAscii</u>	ASCII-Text laden
	<u>WinDocLoadRtf</u>	RTF-Text laden
	<u>WinDocLoadHtml</u>	HTML-Format laden
	<u>WinDocLoadDoc</u>	Word-Dateien im .doc Format laden
int2	<u>WinDocLoadDocX</u>	Word-Dateien im .docx Format laden
	<u>WinDocLoadOEM</u>	OEM-Text
	<u>WinDocLoadMix</u>	laden Text mit Daten mischen
	<u>WinDocLoadInsert</u>	Text in den bestehenden Text einfügen

alpha3 Verschlüsselungscode (optional)

Resultat int Fehlerwert

Siehe Verwandte Befehle,  
WinDocLoadName(),  
WinDocSaveBin(), BinOpen()

Mit diesem Befehl wird der Inhalt eines binären Objekts in ein CtxDocEdit-Objekt geladen. Der Deskriptor des CtxDocEdit-Objektes wird in (obj), der Deskriptor des binären Objekts in (handle1) übergeben.

Der Parameter (int2) bestimmt das zu lesende Format. Folgende Konstanten können angegeben werden:

- WinDocLoadAscii  
Laden von Text im ASCII-Format.
- WinDocLoadRtf  
Laden von Text im RTF-Format.
- WinDocLoadHtml  
Laden von Text im HTML-Format.
- WinDocLoadDoc

## Kontakt

Word-Dateien im .doc Format laden.

- WinDocLoadDocX Word-Dateien im .docx Format laden.

- WinDocLoadOEM

Laden von Text im OEM-Format.

- WinDocLoadMix Text mit Daten mischen.

- WinDocLoadInsert

Der zu ladende Text wird in einen bestehenden Text eingefügt.

Die Parameter zum Quellenformat können mit WinDocLoadInsert kombiniert werden, um in einen bestehenden Text den angegebenen Text einzufügen.

Der Text ersetzt den Text, der mit der Markierung (\$ctxDocEdit->cpiSelLength) selektiert ist. Ist keine Markierung vorhanden, wird der Text an der aktuellen Cursorposition \$ctxDocEdit->cpiSelStart eingefügt.

Bei der Kombination mit der Option WinDocLoadMix werden beim Laden des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.



WinDocLoadMix kann nicht zusammen mit WinDocLoadDoc oder WinDocLoadDocX angegeben werden.

Als Rückgabewert kann neben den Fehlerkonstanten aus dem Bereich der binären Objekte der Wert ErrGeneric zurückgegeben werden, wenn ein interner Fehler aufgetreten ist. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.



Intern wird die Funktion LoadFromMemory der Text-Control-Bibliothek aufgerufen. Dabei wird die Eigenschaft LoadSaveAttribute der Bibliothek beachtet. Die Eigenschaft kann mit \$ctxDocEdit->cpiLoadSaveAttribute gelesen und gesetzt werden. Nähere Informationen finden Sie auf der Hersteller-Seite des Moduls.

Beispiel:

```
// Word-Dokument aus binärem Objekt laden $ctxDocEdit->WinDocLoadBin(tBinFileHdl, _WinDocLoadDoc);
```

Mögliche Laufzeitfehler

ErrHdlInvalid

Bei (obj) handelt es sich nicht um ein CtxDocEdit-Objekt bzw. bei (handle1) nicht um ein BLOb-Objekt.

ErrValueInvalid Argument (int2) enthält ungültige Werte.

## Kontakt

obj -> WinDocLoadName(int1,  
int2[, alpha3]) : int



Text in CtxDocEdit-Objekt laden

obj Objekt (CtxDocEdit-Objekt)

Quelle des Textes

WinStreamNameText internen

int1 Text laden

WinStreamNameFile externen

Text laden

Modus für interne und externe

Texte

WinDocLoadAscii ASCII-Text

laden

WinDocLoadRtf RTF-Text

laden

WinDocLoadHtml HTML-Format

laden

WinDocLoadOEM OEM-Text

laden

WinDocLoadMix Text mit

Daten

mischen

WinDocLoadInsert Text in den

int2 bestehenden

Text einfügen

zusätzlicher Modus für externe

Texte

WinDocLoadDoc Word-Dateien

im .doc

Format laden

WinDocLoadDocX Word-Dateien

im .docx

Format laden

WinDocLoadAuto Textformat

anhand der

Dateiendung

automatisch

erkennen

alpha3 Name des Textes (optional)

Resultat int Fehlerwert

Verwandte Befehle,

Siehe WinDocLoadBin(),

WinDocSaveName()

Mit diesem Befehl wird ein Text in ein CtxDocEdit-Objekt geladen. Der Deskriptor des Objektes wird in (obj) übergeben.

In (int1) wird die Quelle des zu ladenden Textes definiert. Der Name der Quelle wird in (alpha3) angegeben.

## Kontakt

Folgende Quellen können angegeben werden:

- WinStreamNameText

Der Text steht in einem internen Text zur Verfügung. Der Name des Textes wird in (alpha3) übergeben.

- WinStreamNameFile

Der Text steht in einer externen Datei zur Verfügung. Der Name des Textes kann in (alpha3) übergeben werden. Ist er nicht angegeben, wird er der Eigenschaft FileName entnommen.

Der Parameter (int2) bestimmt das Format der Quelle. Folgende Konstanten können übergeben werden:

Modus für interne und externe Texte

- WinDocLoadAscii

Laden von Text im ASCII-Format.

- WinDocLoadRtf

Laden von Text im RTF-Format.

- WinDocLoadHtml

Laden von Text im HTML-Format.

- WinDocLoadOEM

Laden von Text im OEM-Format.

- WinDocLoadMix Text mit

Daten mischen.

- WinDocLoadInsert

Der zu ladende Text wird in einen bestehenden Text eingefügt.

zusätzlicher Modus für externe Texte

- WinDocLoadDoc Word-Dateien im .doc

Format laden.

- WinDocLoadDocX Word-Dateien im .docx

Format laden.

- WinDocLoadAuto

Textformat anhand der Dateiendung automatisch erkennen.

Die Parameter zum Quellenformat können mit WinDocLoadInsert kombiniert werden, um in einen bestehenden Text den angegebenen Text einzufügen.

## Kontakt

Der Text ersetzt den Text, der mit der Markierung (\$ctxDocEdit->cpiSelLength) selektiert ist. Ist keine Markierung vorhanden, wird der Text an der aktuellen Cursorposition \$ctxDocEdit->cpiSelStart eingefügt.

Bei der Kombination mit der Option WinDocLoadMix werden beim Laden des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.



WinDocLoadMix kann nicht zusammen mit WinDocLoadDoc oder WinDocLoadDocX angegeben werden.

Als Rückgabewert kann neben den Fehlerkonstanten aus dem Bereich der externe Dateien der Wert ErrGeneric zurückgegeben werden, wenn ein interner Fehler aufgetreten ist. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.



Intern wird die Funktion LoadFromMemory der Text-Control-Bibliothek aufgerufen. Dabei wird die Eigenschaft LoadSaveAttribute der Bibliothek beachtet. Die Eigenschaft kann mit \$ctxDocEdit->cpiLoadSaveAttribute gelesen und gesetzt werden. Nähere Informationen finden Sie auf der Hersteller-Seite des Moduls.

Beispiel:

```
// externes Word-Dokument laden$ctxDocEdit->WinDocLoadName(_WinStreamNameFile, _WinDocLoadDoc, _S
```

Mögliche Laufzeitfehler

ErrHdlInvalid      Bei (obj) handelt es sich nicht um ein CtxDocEdit-Objekt

ErrValueInvalid Argument (int1) oder (int2) enthält ungültige Werte oder der Wert von (alpha3) ist leer und FileName ist nicht angegeben.

## Kontakt

obj -> WinDocPrint([alpha1[, int2[,  
int3]]) : int



Inhalt des CtxDocEdit-Objektes drucken  
Objekt

obj (CtxDocEdit-Objekt)

alpha1 Seitenauswahl  
(optional)

int2 Deskriptor eines  
PrintDevice-Objektes  
(optional)

int3 Anzahl der exemplare  
(optional)

Resultat int Fehlerwert



Siehe Verwandte Befehle

Mit diesem Befehl wird der Inhalt des CtxDocEdit-Objektes gedruckt. Der Deskriptor des CtxDocEdit-Objektes wird in (obj) übergeben.

Im Parameter (alpha1) kann eine Seitenauswahl definiert werden. Ist dieser Parameter leer, wird der gesamte Inhalt des Objektes gedruckt. Es kann eine der Anweisungen 'range', 'odd' oder 'even' übergeben werden:

- range()

Dieser Eintrag definiert einen Bereich oder eine Aufzählung von Seiten die gedruckt werden soll. In Klammern wird die Aufzählung der Seitennummern, ein Bereich von Seiten oder eine Aufzählung von Seitenbereichen angegeben. Bereichsüberlappungen sind nicht erlaubt und führen zu einem Laufzeitfehler.

Wert	Resultat
'range()'	Alle Seiten
'range(1-10)'	Seiten 1-10
'range(1,3,5-9)'	Seiten 1, 3 und 5-9
'range(1-10,4-7)'	Laufzeitfehler!

- odd()

Dieser Eintrag definiert einen Bereich ungerader Seiten die gedruckt werden sollen.

Wert	Resultat
'odd()'	Alle ungeraden Seiten
'odd(1,10)'	Alle ungeraden Seiten im Bereich 1-10
'odd(1-10)'	Alle ungeraden Seiten im Bereich 1-10

- even()

Dieser Eintrag definiert einen Bereich gerader Seiten die gedruckt werden sollen.

Wert	Resultat
------	----------



## Kontakt

'even()'            Alle geraden Seiten  
'even(1,10)' Alle geraden Seiten im Bereich 1-10  
'even(1-10)' Alle geraden Seiten im Bereich 1-10

Im Parameter (int2) kann der Deskriptor eines PrintDevice-Objektes übergeben werden. Wird dieser Parameter nicht angegeben, erfolgt der Druck auf dem Standarddrucker.



Die Eigenschaften ZoomFactor, QualityX und QualityY vom PrintDevice haben auf diesen Befehl keine Wirkung.

Die Anzahl der zu druckenden Exemplare kann im Parameter (int3) festgelegt werden. Ist der Parameter nicht angegeben, wird die Eigenschaft Copies vom PrintDevice-Objekt ausgewertet. Ist auch diese nicht definiert, wird der Inhalt ein Mal gedruckt.

Beispiel:

```
// Seiten 1, 3 und 5 bis 7 des Dokumentes auf dem Drucker in tPrtDevice 2 Mal drucken$ctxDocEdit-
```

Folgende Fehlerwerte werden von der Funktion zurückgegeben:

ErrGeneric            Allgemeiner Fehler beim Drucken.

ErrOutOfMemory Speicher für die Interpretation der Seitenauswahl oder den Drucker  
konnte nicht angefordert werden.

Mögliche Laufzeitfehler

ErrHdlInvalid        Bei (obj) handelt es sich nicht um ein CtxDocEdit-Objekt bzw. bei (int2) nicht  
um ein PrintDevice-Objekt.

ErrValueInvalid Argument (alpha1) enthält ungültige Werte.

## Kontakt

obj -> WinDocSaveBin(handle1, int2[, int3[, alpha4]]) :



int

Text aus CtxDocEdit-Objekt in binärem Objekt speichern

obj Objekt (CtxDocEdit-Objekt)

handle1 Deskriptor des binären Objekts

Modus	
<u>_WinDocSaveAscii</u>	ASCII-Text sichern
<u>_WinDocSaveRtf</u>	RTF-Text sichern
<u>_WinDocSaveHtml</u>	HTML-Format sichern
<u>_WinDocSaveDoc</u>	Inhalt im .doc Format sichern
<u>_WinDocSaveDocX</u>	Inhalt im .docx Format sichern
<u>_WinDocSavePdf</u>	Inhalt im PDF-Format sichern
<u>_WinDocSaveMix</u>	Text mit Daten mischen
<u>_WinDocSaveOEM</u>	OEM-Text sichern
<u>_WinDocSaveMark</u>	Markierten Bereich sichern
int3	Kompressionsfaktor (optional)
alpha4	Verschlüsselungscode (optional)
Resultat int	Fehlerwert

### Verwandte Befehle,

Siehe WinDocSaveName(),  
WinDocLoadBin(), BinOpen()

Mit diesem Befehl wird ein Text aus einem CtxDocEdit-Objekt in einem binären Objekt gespeichert. Der Deskriptor des CtxDocEdit-Objektes wird in (obj), der Deskriptor des binären Objekts in (handle1) übergeben. Das binäre Objekt (handle1) muss beim Öffnen mit BinLock oder BinSingleLock gesperrt werden.

Der Parameter (int2) bestimmt das zu schreibende Format. Folgende Konstanten können angegeben werden:

- \_WinDocSaveAscii  
ASCII-Text sichern
- \_WinDocSaveRtf

## Kontakt

RTF-Text sichern

- WinDocSaveHtml

HTML-Format sichern

- WinDocSaveDoc

Inhalt im .doc Format sichern

- WinDocSaveDocX

Inhalt im .docx Format sichern

- WinDocSavePdf

Inhalt im PDF-Format sichern

- WinDocSaveOEM

OEM-Text sichern

- WinDocSaveMix Text mit

Daten mischen

- WinDocSaveMark

Markierten Bereich sichern

Die Parameter zum Zielformat können mit WinDocSaveMark kombiniert werden, um einen markierten Teil innerhalb des Textes zu sichern.

Bei der Kombination mit der Option WinDocSaveMix werden beim Speichern des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.



WinDocSaveMix kann nicht zusammen mit WinDocSaveDoc, WinDocSaveDocX oder WinDocSavePdf angegeben werden.

Als Rückgabewert kann neben den Fehlerkonstanten aus dem Bereich der binären Objekte der Wert ErrGeneric zurückgegeben werden, wenn ein interner Fehler aufgetreten ist. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.



Intern wird die Funktion SaveToMemory der Text-Control-Bibliothek aufgerufen. Dabei wird die Eigenschaft LoadSaveAttribute der Bibliothek beachtet. Die Eigenschaft kann mit `$ctxDocEdit->cpiLoadSaveAttribute` gelesen und gesetzt werden. Nähere Informationen finden Sie auf der Hersteller-Seite des Moduls.

Beispiel:

```
// Word-Dokument als binäres Objekt speichern $ctxDocEdit->WinDocSaveBin(tBinFileHdl, _WinDocSaveD
```

Mögliche Laufzeitfehler

ErrHdlInvalid

Bei (obj) handelt es sich nicht um ein CtxDocEdit-Objekt bzw. bei

(handle1) nicht um ein binäres Objekt.

ErrValueInvalid Argument (int2) enthält ungültige Werte.



obj -> WinDocSaveName(int1, int2[,  
alpha3]) : int

Text aus CtxDocEdit-Objekt speichern

obj Objekt (CtxDocEdit-Objekt)

Ziel des Textes

WinStreamNameText Text in  
einen  
internen  
Text  
int1 schreiben

WinStreamNameFile Text in  
eine  
externe  
Datei  
schreiben

Modus für interne und externe

Texte

WinDocSaveAscii ASCII-Text  
sichern

WinDocSaveRtf RTF-Text  
sichern

WinDocSaveHtml HTML-Format  
sichern

WinDocSaveOEM OEM-Text  
sichern

WinDocSaveMix Text mit  
Daten  
mischen

WinDocSaveMark Markierten  
Bereich  
int2 sichern

zusätzlicher Modus für externe  
Texte

WinDocSaveDoc Inhalt im .doc  
Format  
sichern

WinDocSaveDocX Inhalt im  
.docx Format  
sichern

WinDocSavePdf Inhalt im  
PDF-Format  
sichern

WinDocSaveAuto Textformat  
anhand der  
Dateiendung  
automatisch  
erkennen

alpha3 Zielname

Resultat int Fehlerwert

Verwandte Befehle,

Siehe WinDocSaveBin(),  
WinDocLoadName()

Mit diesem Befehl wird ein Text aus einem CtxDocEdit-Objekt gespeichert. Der Deskriptor des Objektes wird in (obj) übergeben.

In (int1) wird das Ziel des Textes definiert. Der Name des Ziels wird in (alpha3) angegeben.

Folgende Ziele können angegeben werden:

- WinStreamNameText

Der Text wird in einem internen Text gespeichert. Der Name des Textes wird in (alpha3) übergeben.

- WinStreamNameFile

Der Text wird in einer externen Datei gespeichert. Der Name des Textes kann in (alpha3) übergeben werden. Ist er nicht angegeben, wird er der Eigenschaft FileName entnommen.

Der Parameter (int2) bestimmt das zu schreibende Format. Folgende Konstanten können übergeben werden:

Modus für interne und externe Texte

- WinDocSaveAscii

ASCII-Text sichern

- WinDocSaveRtf

RTF-Text sichern

- WinDocSaveHtml

HTML-Format sichern

- WinDocSaveOEM

OEM-Text sichern

- WinDocSaveMix Text mit

Daten mischen

- WinDocSaveMark Markierten

Bereich sichern

zusätzlicher Modus für externe Texte

- WinDocSaveDoc

Inhalt im .doc Format sichern

## Kontakt

- WinDocSaveDocX

Inhalt im .docx Format sichern

- WinDocSavePdf

Inhalt im PDF-Format sichern

- WinDocSaveAuto

Textformat anhand der Dateieindung automatisch erkennen

Die Parameter zum Zielformat können mit WinDocSaveMark kombiniert werden, um einen markierten Teil innerhalb des Textes zu sichern.

Bei der Kombination mit der Option WinDocSaveMix werden beim Speichern des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.



WinDocSaveMix kann nicht zusammen mit WinDocSaveDoc, WinDocSaveDocX oder WinDocSavePdf angegeben werden.

Als Rückgabewert kann neben den Fehlerkonstanten aus dem Bereich der externe Dateien der Wert ErrGeneric zurückgegeben werden, wenn ein interner Fehler aufgetreten ist. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.



Intern wird die Funktion SaveToMemory der Text-Control-Bibliothek aufgerufen. Dabei wird die Eigenschaft LoadSaveAttribute der Bibliothek beachtet. Die Eigenschaft kann mit `$ctxDocEdit->apiLoadSaveAttribute` gelesen und gesetzt werden. Nähere Informationen finden Sie auf der Hersteller-Seite des Moduls.

Beispiel:

```
// Dokument extern als PDF speichern$ctxDocEdit->WinDocSaveName(_WinStreamNameFile, _WinDocSavePd
```

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein CtxDocEdit-Objekt

ErrValueInvalid Argument (int1) oder (int2) enthält ungültige Werte oder der Wert von (alpha3) ist leer und FileName ist nicht angegeben.

## Kontakt

obj -> WinDocUserDictAddName(alpha1) : int  CtxDocEdit-Objekt ein

benutzerdefiniertes Wörterbuch hinzufügen

obj        Objekt (CtxDocEdit-Objekt)

alpha1    Pfad und Name des Wörterbuchs

Fehlercode

ErrOk

Wörterbuch hinzugefügt

Resultat int ErrGeneric

CtxDocEdit-Erweiterung nicht

installiert

ErrFsiNoFile Wörterbuch nicht vorhanden

Verwandte Befehle, WinDocUserDictRemoveName(),

Siehe

Blog

Fügt dem CtxDocEdit-Objekt (obj) ein benutzerdefiniertes Wörterbuch hinzu. Das Wörterbuch muss im .tlx-Format gespeichert sein (siehe Aufbau einer tlx-Datei).

In (alpha1) steht der Pfad zu der .tlx-Datei. Sofern die .tlx-Datei im Verzeichnis der wspell.ocx liegt, genügt der Dateiname. Die wspell.ocx liegt standardmäßig im Installationspfad von CONZEPT 16 unter Common\CtxDocEdit\Spell\.

Beim Hinzufügen wird geprüft, ob die Datei existiert. Wenn die Datei nicht existiert kann sie auch nicht dem CtxDocEdit-Objekt hinzugefügt werden. In dem Fall wird ErrFsiNoFile zurückgegeben. Ein ungültiges Dateiformat wird nicht erkannt.

Die CtxDocEdit-Erweiterung muss installiert sein (siehe InstallCtxDocEdit).

Die Wörterbücher werden intern kommasepariert in einer Eigenschaft gespeichert, die Eigenschaft hat eine Maximallänge von 1023 Zeichen, wird diese überschritten wird der Laufzeitfehler ErrStringOverflow ausgelöst.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:

ErrOk

Das Wörterbuch wurde dem CtxDocEdit-Objekt (obj) hinzugefügt.

ErrGeneric

Die CtxDocEdit-Erweiterung (siehe InstallCtxDocEdit) ist nicht installiert.

ErrFsiNoFile Die Datei (alpha1) existiert nicht.

Beispiele:

```
// Datei liegt in C:\MyDictionaries$DocEdit->WinDocUserDictAddName('C:\MyDictionaries\Example.tlx
```

Mögliche Laufzeitfehler:

ErrHdlInvalid

Der Deskriptor des CtxDocEdit-Objektes (obj) ist ungültig.

ErrStringOverflow

Dem CtxDocEdit-Objekt (obj) wurden zu viele Wörterbücher hinzugefügt.

## Kontakt



**obj -> WinDocUserDictRemoveName(alpha1) Benutzerdefiniertes  
Wörterbuch aus CtxDocEdit-Objekt entfernen**

obj  
Objekt

**(CtxDocEdit-Objekt)**

**alpha1** Pfad und Name des  
Wörterbuchs

**Verwandte Befehle,**

Siehe WinDocUserDictAddName(),

Blog

Entfernt ein benutzerdefiniertes Wörterbuch aus dem CtxDocEdit-Objekt.

In (alpha1) steht der Pfad zu der .tlx-Datei. Sofern die .tlx-Datei im Verzeichnis der wspell.ocx liegt, genügt der Dateiname. Die wspell.ocx liegt standardmäßig im Installationspfad von CONZEPT 16 unter Common\CtxDocEdit\Spell\.

Die CtxDocEdit-Erweiterung muss installiert sein (siehe InstallCtxDocEdit).

Die Funktion besitzt keinen Rückgabewert.

**Beispiele:**

```
// Datei liegt in C:\MyDictionaries$DocEdit->WinDocUserDictRemoveName('C:\MyDictionaries\Example.
```

**Mögliche Laufzeitfehler:**

**ErrHdlInvalid** Der Deskriptor des CtxDocEdit-Objektes (obj) ist ungültig.



Konstanten für CtxDocEdit-Befehle

Konstanten für CtxDocEdit-Befehle

Siehe CtxDocEdit-Befehle

- WinDocLoadAscii
- WinDocLoadAuto
- WinDocLoadDoc
- WinDocLoadDocX
- WinDocLoadHtml
- WinDocLoadInsert
- WinDocLoadMix
- WinDocLoadOEM
- WinDocLoadRtf
- WinDocSaveAscii
- WinDocSaveAuto
- WinDocSaveDoc
- WinDocSaveDocX
- WinDocSaveHtml
- WinDocSaveMark
- WinDocSaveMix
- WinDocSaveOEM
- WinDocSavePdf
- WinDocSaveRtf
- WinStreamNameFile
- WinStreamNameText

**\_WinDocLoadAscii**

**Text im ASCII-Format laden**

**Wert 2 / 0x0002**

**Siehe WinDocLoadName(),**

**Der Text wird im ASCII-Format geladen.**

## Kontakt

**\_WinDocLoadAuto**

**Format automatisch bestimmen**

**Wert 0 / 0x0000**

**Siehe WinDocLoadName(),**

**Das Format des zu ladenden Textes wird automatisch anhand der Dateiendung bestimmt.  
Dies funktioniert nur beim Laden von externen Texten.**

**\_WinDocLoadDoc**

**Text im .doc-Format laden**

**Wert 512 / 0x0200**

**Siehe WinDocLoadName(),**

**Der Text wird im .doc-Format geladen.**

**\_WinDocLoadDocX**

**Text im .docx-Format laden**

**Wert 1.024 / 0x0400**

**Siehe WinDocLoadName(),**

**Der Text wird im .docx-Format geladen.**

**\_WinDocLoadHtml**

**Text im HTML-Format laden**

**Wert 256 / 0x0100**

**Siehe WinDocLoadName(),**

**Der Text wird im HTML-Format geladen.**

## Kontakt

**\_WinDocLoadInsert**

**Text einfügen**

**Wert 16 / 0x0010**

**Siehe WinDocLoadName(),**

**Der zu ladende Text wird in einen bestehenden Text eingefügt.**

## Kontakt

**\_WinDocLoadMix**

**Text mit Daten beim Laden mischen**

**Wert 8 / 0x0008**

**Siehe WinDocLoadName(),**

**Beim Laden des Textes werden die enthaltenen Platzhalter durch den entsprechenden Inhalt ersetzt. Die Platzhalter sind mit Markierungszeichen geklammert, die in der Eigenschaft DocMixMarker des \_App-Objektes definiert werden kann.**

**Weitere Hinweise befinden sich im Abschnitt Text und Daten mischen.**



**\_WinDocLoadOEM**

**Text im OEM-Format laden**

**Wert 4 / 0x0004**

**Siehe WinDocLoadName(),**

**Der Text wird im OEM-Format geladen.**

**\_WinDocLoadRtf**

**Text im RTF-Format laden**

**Wert 1 / 0x0001**

**Siehe WinDocLoadName(),**

**Der Text wird im RTF-Format geladen.**

**\_WinDocSaveAscii**

**Speichern im ASCII-Format**

**Wert 2 / 0x0002**

**Siehe WinDocSaveName(),**

**Der Text wird im ASCII-Format gespeichert. Die Formatierungen gehen dabei verloren.**

## Kontakt

**\_WinDocSaveAuto**

**Format automatisch bestimmen**

**Wert 0 / 0x0000**

**Siehe WinDocSaveName()**

**Das Format wird automatisch anhand der Dateiendung bestimmt. Dazu muss die Eigenschaft FileName gesetzt sein.**

**\_WinDocSaveDoc**

**Speichern im .doc-Format**

**Wert 512 / 0x0200**

**Siehe WinDocSaveName(),**

**Der Text wird im .doc-Format gespeichert.**

**\_WinDocSaveDocX**

**Speichern im .docx-Format**

**Wert 1.024 / 0x0400**

**Siehe WinDocSaveName(),**

**Der Text wird im .docx-Format gespeichert.**

**\_WinDocSaveHtml**

**Speichern im HTML-Format**

**Wert 256 / 0x0100**

**Siehe** **WinDocSaveName()**,

**Der Text wird im HTML-Format gespeichert.**

**\_WinDocSaveMark**

**Markierung speichern**

**Wert 16 / 0x0010**

**Siehe WinDocSaveName(),**

**Der markierte Bereich wird gespeichert.**



## Kontakt

**\_WinDocSaveMix**

**Text mit Daten beim Laden mischen**

**Wert 8 / 0x0008**

**Siehe WinDocSaveName(),**

**Beim Speichern des Textes werden die enthaltenen Platzhalter durch den entsprechenden Inhalt ersetzt. Die Platzhalter sind mit Markierungszeichen geklammert, die in der Eigenschaft DocMixMarker des \_App-Objektes definiert werden können.**

**Weitere Hinweise befinden sich im Abschnitt Text und Daten mischen.**

## Kontakt

**\_WinDocSaveOEM**

**Speichern im OEM-Format**

**Wert 4 / 0x0004**

**Siehe WinDocSaveName(),**

**Der Text wird im OEM-Format gespeichert. Die Formatierungen gehen dabei verloren.**

**\_WinDocSavePdf**

**Speichern im PDF-Format**

**Wert 2.048 / 0x0800**

**Siehe WinDocSaveName(),**

**Der Text wird im PDF-Format gespeichert.**

**\_WinDocSaveRtf**

**Speichern im RTF-Format**

**Wert 1 / 0x0001**

**Siehe** **WinDocSaveName()**,

**Der Text wird im RTF-Format gespeichert.**

**\_WinStreamNameFile**

**Quelle des Textes**

**Wert 5**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamNameFile gesetzt, wird der Inhalt der in der Eigenschaft FileName angegebenen externen Datei dargestellt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad(), WinRtfLoadName(), WinRtfSave(), WinRtfSaveName(), WinDocLoadName() und WinDocSaveName() anzugeben.**

**\_WinStreamNameText**

**Quelle des Textes**

**Wert 4**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamNameText gesetzt, wird der in der Eigenschaft FileName angegebene interne Text dargestellt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad(), WinRtfLoadName(), WinRtfSave(), WinRtfSaveName(), WinDocLoadName() und WinDocSaveName() anzugeben.**

### DataList-Befehle

Befehle zum Bearbeiten eines DataList-Objekts

Verwandte  
Befehle,

Siehe DataList,  
Befehlsgruppen,  
Befehlsliste

### Befehle

- WinLstCellGet
- WinLstCellSet
- WinLstDatLineAdd
- WinLstDatLineInfo
- WinLstDatLineRemove
- WinLstEdit

### Konstanten

- WinLstDatInfoCount
- WinLstDatLineAll
- WinLstDatLineCurrent
- WinLstDatLineLast
- WinLstDatLineSelected
- WinLstDatModeDefault
- WinLstDatModeSortInfo

obj ->

WinLstDatLineInfo(int1) 

: int

Zeilen-Informationen

obj      Objekt

Option

WinLstDatInfoCount Anzahl

int1

der

Zeilen

Resultat int Zeilen-Information

DataList-Befehle,

Siehe

Verwandte Befehle

Wird WinLstDatInfoCount übergeben, liefert dieser Befehl die Anzahl der Zeilen zurück, die sich momentan in dem Objekt DataList befinden.

Mögliche Laufzeitfehler:

ErrHdlInvalid DataList (obj) ungültig



obj ->

WinLstDatLineAdd(var1[, )

int2]) : int

Zeile hinzufügen

obj      Objekt

var1      Daten

Zeile (optional)

WinLstDatLineCurrent eine Zeile

wird  
hinter der  
aktuellen  
Zeile

int2

WinLstDatLineLast

eingefügt  
eine Zeile  
wird am  
Ende  
angefügt  
(default)

Resultat int      Zeilennummer

WinLstDatLineRemove(),

Siehe DataList-Befehle, Verwandte Befehle

Mit diesem Befehl werden dem Objekt DataList Zeilen hinzugefügt. Mit (var1) wird die erste Zelle der Zeile direkt gesetzt. (var1) kann von verschiedenem Typ sein, muss aber mit der Eigenschaft ClmType der 1. Spalte übereinstimmen. Als Zeile (int2) kann entweder die Zeilennummer direkt angegeben, oder eine der beiden definierten Konstanten verwendet werden.

Als Resultat wird die neue Zeilennummer zurückgeliefert.


Beispiel:

```
WinOpen('FrmDataList', _WinOpenDialog);tDataList # $DataList;for    tLine # 1;loop    Inc(tLine);unt
```

Mögliche Laufzeitfehler:

ErrHdlInvalid DataList (obj) ungültig

obj ->

WinLstDatLineRemove(int1) 

: logic

Zeile löschen

obj      Objekt

Zeile

\_WinLstDatLineCurrent

Aktuelle  
Zeile wird  
gelöscht

\_WinLstDatLineLast

Letzte  
Zeile wird

int1

\_WinLstDatLineAll

gelöscht  
Alle Zeilen  
werden

\_WinLstDatLineSelected

gelöscht  
Selektierte  
Zeilen  
werden  
gelöscht

Resultat logic Funktion erfolgreich  
durchgeführt

WinLstDatLineAdd(),

Siehe DataList-Befehle, Verwandte  
Befehle

Mit diesem Befehl kann eine oder mehrere Zeilen des Objekts DataList gelöscht werden.

Sollen mit der Option \_WinLstDatLineSelected mehrere Zeilen gelöscht werden, muss beim DataList-Objekt die Mehrfachauswahl (MultiSelect) aktiviert sein.

Wurde die Funktion erfolgreich durchgeführt ist das Ergebnis true andernfalls false.


Beispiel

```
// DataList-Objekt leerentHdlData->WinLstDatLineRemove(_WinLstDatLineAll); // Selektierte Zeilen 1
```

Mögliche Laufzeitfehler:

ErrHdlInvalid DataList (obj) ist ungültig

obj ->

WinLstCellGet(var1[,  int2[, int3[, int4]]) :

logic

Spalte lesen

obj Objekt

var1 Daten

int2 Spalte (optional)

Zeile (optional)

Nummer  
der zu  
lesenden

Zeile

int3

WinLstDatLineCurrent Aktuelle

Zeile wird  
gelesen

WinLstDatLineLast

Letzte  
Zeile wird  
gelesen

(default)

Modus (optional)

WinLstDatModeDefault Anzeigewerte

int4

ermitteln  
(default)

WinLstDatModeSortInfo Sortiertwerte  
ermitteln

Resultat logic

Funktion erfolgreich  
durchgeführt

Verwandte Befehle,

Siehe WinLstCellSet(), DataList-Befehle,  
Blog

Mit diesem Befehl wird der Inhalt oder der Sortierwert einer Spalte des Objekts DataList gelesen.

In (var1) wird der Inhalt oder der Sortierwert der angegebene Spalte übertragen. Die Variable (var1) kann von verschiedenem Typ sein, muss aber mit der Eigenschaft ClmType für den Inhalt bzw. mit der Eigenschaft ClmTypeSort für den Sortierwert der angegebenen Spalte übereinstimmen.

In (int2) kann die Position der Spalte angegeben werden. Die Position lässt sich ausgehend vom Deskriptor der Spalte über WinInfo() mit der Option \_WinItem ermitteln.

Bei der Zeile (int3) kann entweder die Zeile direkt, oder eine der beiden Konstanten angegeben werden. In Verbindung mit den Ereignissen EvtKeyItem, EvtMouseItem und EvtLstDataInit kann in (int3) der Parameter aID angegeben werden, da in dieser Variable die Zeilennummer steht.

Im Argument (int4) kann der Abfragemodus angegeben werden.

## Kontakt

Wurde die Funktion erfolgreich durchgeführt ist das Ergebnis true andernfalls false.

Beispiele:

```
// Erste Spalte der aktuellen Zeile lesentHdlDataList->WinLstCellGet(tCellContent); // Zweite Spalte
```

Mögliche Laufzeitfehler:

ErrHdlInvalid DataList (obj) ungültig

obj ->

WinLstCellSet(var1[,  
int2[, int3[, int4]]) :



logic

Spalte setzen

obj      Objekt

var1      Daten

int2      Spalte (optional)

Zeile (optional)

\_WinLstDatLineCurrent Aktuelle

Zeile wird

int3

\_WinLstDatLineLast

geändert

Letzte

Zeile wird

geändert

(default)

Modus (optional)

\_WinLstDatModeDefault Anzeigewerte

int4

definieren

(default)

\_WinLstDatModeSortInfo Sortiertwerte

definieren

Resultat logic      Funktion erfolgreich

durchgeführt

Verwandte Befehle,

Siehe WinLstCellGet(), DataList-Befehle,

Blog

Mit diesem Befehl wird der Inhalt oder der Sortierwert einer Spalte des Objekts DataList gesetzt. Mit (var1) wird die angegebene Spalte der Zeile gesetzt. (var1) kann von verschiedenem Typ sein, muss aber mit der Eigenschaft ClmType für den Inhalt bzw. mit der Eigenschaft ClmTypeSort für den Sortierwert der angegebenen Spalte übereinstimmen.

In (int2) kann die Position der Spalte angegeben werden. Die Position lässt sich ausgehend vom Deskriptor der Spalte über WinInfo() mit der Option \_WinItem ermitteln.

Bei der Zeile (int3) kann entweder die Zeile direkt, oder eine der beiden Konstanten angegeben werden.

Im Argument (int4) kann der Einfügemodus angegeben werden. Ist die Eigenschaft ClmTypeSort bei einer Spalte gesetzt, sollten für jede Zeile der Spalte auch Sortierwerte definiert werden, da das Ergebnis der Sortierung sonst nicht definiert ist.

Wurde die Funktion erfolgreich durchgeführt ist das Ergebnis true andernfalls false.

Beispiel:

```
// Datentyp für SortiervergleichsClm->wpClmTypeSort # _TypeBigInt;...// Sortierungswert für Spalt
```

Mögliche Laufzeitfehler:

ErrHdlInvalid DataList (obj) ungültig

Konstanten für DataList-Befehle

Konstanten für DataList-Befehle

Siehe DataList-Befehle

Konstanten

- WinLstDatInfoCount
- WinLstDatLineAll
- WinLstDatLineCurrent
- WinLstDatLineLast
- WinLstDatLineSelected
- WinLstDatModeDefault
- WinLstDatModeSortInfo

## Kontakt

**\_WinLstDatInfoCount**

Anzahl Zeilen

Wert 1

**Siehe** Verwandte

Befehle

Option bei WinLstDatLineInfo(). Es wird die Anzahl der Zeilen, die sich in der Liste befinden, zurückgegeben.



## Kontakt

**\_WinLstDatLineAll**

Alle Zeilen

Wert -3

**Siehe** Verwandte  
Befehle

Option bei WinLstDatLineRemove(). Es werden alle Zeilen gelöscht.

## Kontakt

\_WinLstDatLineCurrent

Bedeutung je nach Befehl

Wert -1

**Siehe** Verwandte

Befehle

Option bei folgenden Befehlen:

WinLstDatLineAdd() Zeile wird hinter der aktuellen Zeile angefügt

WinLstDatLineRemove() aktuelle Zeile wird gelöscht

WinLstCellSet() aktuelle Zeile wird geändert

WinLstCellGet() aktuelle Zeile wird gelesen

## Kontakt

**\_WinLstDatLineLast**

**Bedeutung je nach Befehl**

**Wert -2**

**Siehe** Verwandte  
Befehle

**Option bei folgenden Befehlen:**

**WinLstDatLineAdd()**      Zeile wird am Ende angefügt

**WinLstDatLineRemove()** letzte Zeile wird gelöscht

**WinLstCellSet()**      letzte Zeile wird geändert

**WinLstCellGet()**      letzte Zeile wird gelesen

## Kontakt

\_WinLstDatLineSelected

Selektierte Zeilen löschen

Wert -4

**Siehe** Verwandte

Befehle

Option bei WinLstDatLineRemove() - Sofern bei dem DataList-Objekt die Eigenschaft MultiSelect gesetzt ist, werden alle selektierten Datensätze des DataList-Objektes entfernt.

**\_WinLstDatModeDefault**

Daten angeben / abfragen

Wert 0

### Verwandte

Siehe Befehle,

WinLstCellGet(),

WinLstCellSet()

Modus bei WinLstCellGet() und WinLstCellSet(), mit dem die Daten einer Zelle ermittelt und gesetzt werden können. Dies ist der Standardmodus.

**\_WinLstDatModeSortInfo**

Sortierwert angeben / abfragen

Wert 1

### Verwandte

Siehe Befehle,

WinLstCellGet(),

WinLstCellSet()

Modus bei WinLstCellGet() und WinLstCellSet(), mit dem der Sortierwert einer Zelle ermittelt und gesetzt werden kann.

### Dialog-Befehle

Befehle zum Laden/Anzeigen von Dialog-Objekten

Verwandte  
Befehle,

Siehe

Befehlsgruppen,  
Befehlsliste

Befehle

- WinAdd
- WinAddByName
- WinClose
- WinDialog
- WinDialogBox
- WinDialogResult
- WinDialogRun
- WinOpen
- WinSave
- WinUrmDialog









Konstanten

- WinAddHidden
- WinDialogAlwaysOnTop
- WinDialogApp
- WinDialogAsync
- WinDialogBoxDefault
- WinDialogBoxUseFont
- WinDialogBoxUseFontButton
- WinDialogBoxUseTextButton
- WinDialogCenter
- WinDialogCenterScreen
- WinDialogCenterScreenX
- WinDialogCenterScreenY
- WinDialogCenterX
- WinDialogCenterY
- WinDialogCreateHidden
- WinDialogMaximized
- WinDialogMinimized
- WinDialogNoActivate
- WinDialogOK
- WinDialogOKCancel
- WinDialogUtf8
- WinDialogYesNo
- WinDialogYesNoCancel
- WinIcoApplication
- WinIcoError
- WinIcoExternFile
- WinIcoInformation
- WinIcoQuestion
- WinIcoWarning
- WinOpenDialog

- WinOpenEventsOff
- WinOpenLock
- WinOpenUnicode
- WinUpdScrollPos
- WinUrmFlagHelpTipOnSysProps



## Kontakt

obj -> WinAdd(handle1[, int2[, handle3]]) : int         Oberflächenobjekt in  
ein anderes Oberflächenobjekt laden

obj Elternojekt

handle1 einzufügendes Oberflächenobjekt

Optionen (optional)

WinAddHidden

MDI-Fenster

unsichtbar

laden

WinDialogMaximized MDI-Fenster

int2

maximiert

laden

WinDialogMinimized MDI-Fenster

minimiert

laden

handle3 Deskriptor des nachfolgenden

Objektes (optional)

Laderesultat

**Resultat** int

ErrOk Laden erfolgreich



Verwandte Befehle,

WinAddByName(), WinCreate(),

Siehe

WinRemove(), WinDestroy(),

Ereignisabläufe MdiFrame

Der Befehl fügt dem angegebenen Elternojekt (obj) das Oberflächenobjekt (handle1) hinzu. Bei dem Oberflächenobjekt muss es sich um einen MdiFrame oder um ein mit WinCreate() angelegtes Objekt handeln. Das Objekt (handle1) kann bereits weitere mit WinCreate() und WinAdd() untergeordnete Oberflächenobjekte enthalten.

Ist das Objekt (handle1) ein MdiFrame, kann über den optionalen Parameter (int2) mit der Konstanten WinAddHidden angegeben werden, dass das MdiFrame-Objekt unsichtbar geladen werden soll. In diesem Fall muss das MdiFrame-Objekt später mit dem Befehl WinUpdate() mit dem Parameter WinUpdOn gezeichnet werden. Mit den Optionen WinDialogMaximized bzw. WinDialogMinimized kann der MdiFrame maximiert oder minimiert werden.

Sind dem MDI-Fenster über die Eigenschaft DbRecBuf eigene Feldpuffer zugeordnet worden, stehen diese nach diesem Befehl zur Verfügung und können initialisiert werden.

Wurde das Oberflächenobjekt (handle1) mit WinCreate() erstellt, kann das nachfolgende Objekt im Argument (handle3) angegeben und somit die Objektreihenfolge definiert werden.



Das Objekt (handle1) darf zuvor nicht bereits mit WinAdd() zu einem anderen Objekt hinzugefügt werden, außer es wurde anschließend mit WinRemove() wieder entfernt.

Bei dem Oberflächenobjekt (handle1) kann es sich nicht um ein Frame-, AppFrame-oder TrayFrame-Objekt handeln. Um diese zu starten, muss, auch für dynamisch erstellte, weiterhin der Befehl WinDialogRun() verwendet werden.

## Kontakt

Wird das Elternobjekt (obj) bereits angezeigt, erfolgt auch die Darstellung des hinzugefügten Oberflächenobjektes (handle1) direkt nach der Ausführung des Befehls.

Als Resultat kann der Fehlerwert ErrType zurückgegeben werden, wenn das hinzuzufügende Objekt (handle1) nicht in das Elternobjekt (obj) eingefügt werden kann. Das Resultat ist ErrExists, wenn das hinzuzufügende Objekt (handle1) bereits einem Objekt hinzugefügt wurde. Konnte ein MdiFrame nicht zu einem AppFrame hinzugefügt werden, ist das Resultat ErrOutOfMemory. Das Resultat ist ErrUnavailable, wenn eine Spalte vor einer bestehenden Spalte eingefügt wird und die Liste bereits Inhalt besitzt. Ist kein Fehler aufgetreten, wird ErrOk zurückgegeben.

### Hinweise für Spalten-Objekte

Dynamische Spalten-Objekte können auch einer nicht dynamisch erstellten DataList oder RecList hinzugefügt werden. Diese darf auch bereits nicht dynamisch erstellte Spalten enthalten.

Ein Spalten-Objekt besitzt eine Anzeigeposition und eine Indexposition. Die Anzeigeposition definiert, wo die Spalte innerhalb der Liste angezeigt wird und kann sich z. B. durch Benutzerinteraktionen mit der Spalte (z. B. Verschieben der Spalte durch den Anwender) ändern. Die Anzeigeposition kann durch die Eigenschaft ClnOrder gesetzt oder auch abgefragt werden.

Die Indexposition definiert, welche Position die Zelle im Datensatz (Zeile) besitzt. Diese Position wird den Befehlen WinLstCellSet() und WinLstCellGet() übergeben, wenn die Daten einer Zelle gesetzt oder abgefragt werden sollen. Die Indexposition kann mit dem Befehl WinInfo() und der Option WinItem ermittelt werden. Bei WinAdd() unter Angabe einer nachfolgenden Spalte erhält die neu hinzugefügte Spalte die die Indexposition der nachfolgenden Spalte. Die Anzeigeposition wird über die Eigenschaft ClnOrder definiert.



Die Angabe eines nachfolgenden Spalten-Objektes ist nicht zulässig, wenn die Liste bereits einen Inhalt besitzt (Rückgabewert ErrUnavailable). Es können jedoch Spalten am Ende eingefügt werden (WinAdd() ohne nachfolgendes Objekt).

Nachdem ein Spalten-Objekt einer DataList hinzugefügt wurde, hat diese zunächst keinen Inhalt. Mit dem Befehl WinLstCellSet() kann der Spalteninhalt entsprechend des durch ClnType definierten Spalten-Typs gesetzt werden. Der Befehl WinLstCellGet() liefert den Wert false, wenn die Zelle der Spalte noch keinen Inhalt besitzt.

### Hinweise für GroupColumn-Objekte

Dynamische GroupColumn-Objekte können auch einem nicht dynamisch erstellten RecView- oder einem nicht dynamisch erstellten, übergeordneten GroupColumn-Objekt hinzugefügt werden.

Ein GroupColumn-Objekt besitzt eine Anzeige- und eine Indexposition. Die Anzeigeposition definiert, wo das GroupColumn-Objekt im RecView bzw. im

## Kontakt

übergeordneten GroupColumn-Objekt angezeigt wird. Die Anzeigeposition kann durch die Eigenschaft VisibleOrder gesetzt und abgefragt werden.

Die Indexposition definiert eine eindeutige fortlaufende Nummer für die Eigenschaft SelectorItem bzw. SelectorSubItem.

Die Angabe eines nachfolgenden GroupColumn-Objektes ist nicht zulässig, wenn das RecView bereits einen Inhalt besitzt. Es können jedoch GroupColumn-Objekte am Ende eingefügt werden (WinAdd() ohne nachfolgendes Objekt).

Wird das RecView-Objekt angezeigt, während WinAdd() für ein GroupColumn-Objekt durchgeführt wird, dann hat dies zur Folge, dass das Ereignis EvtLstGroupInit aufgerufen wird.

### Hinweise zum PopupList-Objekt

Das Objekt kann den Eingabeobjekten (Edit, IntEdit, ...) hinzugefügt werden. Dem PopupList-Objekt kann wiederum ein DataListPopup, RecListPopup- oder StoListPopup-Objekt hinzugefügt werden.

Einem GroupColumn-Objekt kann ein weiteres GroupColumn-Objekt untergeordnet werden. Dies ist jedoch nur zulässig, wenn das übergeordnete Objekt nicht bereits einem GroupColumn-Objekt untergeordnet ist und das unterzuordnende GroupColumn-Objekt seinerseits keine untergeordneten GroupColumn-Objekte enthält.

### Beispiele:

```
// MDI-Frame 'Addresses' laden tMdiFrame # WinOpen('Addresses');// Wenn hinzufügen des MDI-Fensters
```

### Mögliche Laufzeitfehler:

#### ErrHdlInvalid

Elternobjekt (obj) oder hinzuzufügendes Oberflächenobjekt (handle1) ungültig oder das Oberflächenobjekt (handle1) ist nicht dynamisch erstellt worden.

#### ErrMemExhausted

Fenstererstellung ist fehlgeschlagen.  
Das nachfolgende Objekt (handle3) ist angegeben, jedoch kein Oberflächenobjekt oder kein Kindobjekt von dem angegebenen Elternobjekt (obj).

#### ErrIllegalOp

Das hinzuzufügende Objekt (handle1) ist ein Frame, MdiFrame oder AppFrame und ein nachfolgendes Objekt (handle3) ist angegeben.

## Kontakt

obj -> WinAddByName(alpha1[, int2]) : int  MDI-Fenster über

Name in Applikations-Fenster laden

obj Applikations-Fenster

alpha1 MDI-Fenstername

Optionen (optional)

WinAddHidden

MDI-Fenster

unsichtbar

laden

WinDialogMaximized MDI-Fenster

int2

maximiert

laden

WinDialogMinimized MDI-Fenster

minimiert

laden

Resultat int MDI-Fenster

Verwandte Befehle, WinAdd(),

Siehe

Ereignisabläufe MdiFrame

Der Befehl lädt das MdiFrame-Objekt (alpha1) in den Ausgabebereich des AppFrame-Objekts (obj). Über den optionalen Parameter (int2) kann über die Konstante WinAddHidden angegeben werden, ob das MdiFrame-Objekt unsichtbar geladen werden soll. In diesem Fall muss das MdiFrame-Objekt später mit dem Befehl WinUpdate() mit dem Parameter WinUpdOn gezeichnet werden.

Je nach verwendetem Betriebssystem stehen unterschiedlich viele Ressourcen (Benutzer- und GDI-Objekte) zur Darstellung des Fensters zur Verfügung. Ab dem Betriebssystem Windows 2000 wird nach dem Laden des Fensters die verbleibenden Ressourcen überprüft. Stehen weniger als 10% zur Verfügung, wird der Dialog nicht geladen. Die Anweisung gibt 0 als Deskriptor zurück und der globale Fehlerwert wird auf ErrOutOfMemory gesetzt. Die zur Verfügung stehenden Benutzer- und GDI-Objekte können über den Info-Dialog und die Eigenschaften ObjectsUserLimit und ObjectsGDILimit ermittelt werden.

Sind dem MDI-Fenster über die Eigenschaft DbRecBuf eigene Feldpuffer zugeordnet worden, stehen diese nach diesem Befehl zur Verfügung und können initialisiert werden.

Als Rückgabewert wird der Deskriptor des MDI-Fensters zurückgegeben. Konnte das MdiFrame-Objekt nicht erfolgreich zum AppFrame-Objekt hinzugefügt werden, ist das Resultat 0.

Beispiel:

```
// MDI-Fenster 'MdiFrame' in Applikations-Fenster $MdiApp hinufügengtMdi # $MdiApp->WinAddByName('
```

Mögliche Laufzeitfehler:

ErrStringOverflow MDI-Fenstername (alpha1) länger als 40 Zeichen

obj ->

WinClose() :



logic

Fenster entladen

obj Fenster

Resultat logic Entladeerfolg

Verwandte

Befehle,

Siehe

WinOpen(),

WinDialog()

Mit diesem Befehl wird ein Fenster, das mit WinOpen() oder WinDialog() geladen wurde, geschlossen. Als (obj) wird der Deskriptor des zu schließenden Fensters übergeben.

Ein mit WinOpen() geöffneter und mit WinDialogRun() aufgerufenes Fenster kann entweder durch den Benutzer (zum Beispiel durch Drücken der Schließen-Schaltfläche) oder prozedural durch Ausführung des Befehls WinClose() geschlossen werden. In beiden Fällen befindet sich das Fenster noch im Speicher und es kann auf die Eigenschaften der enthaltenen Objekte zugegriffen werden. Das Fenster wird erst durch die erneute Ausführung von WinClose() entladen.

Beispiele:

```
// Fenster 'Message' ladenWinOpen('Message');...// Fenster $Message entladen$Message->WinClose();
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Fenster (obj) ungültig

## Kontakt

WinDialog(alpha1[, int2[,  
handle3]]) : int



Fenster laden und ausführen

alpha1 Fenstername

int2 Optionen (optional; siehe Text)

handle3 Elternfenster (optional)

Schaltflächen-ID

\_WinIdClose Schließen-Schaltfläche

gedrückt

\_WinIdOk OK-Schaltfläche

gedrückt

Resultat int \_WinIdCancel Abbrechen-Schaltfläche

gedrückt

\_WinIdYes Ja-Schaltfläche

gedrückt

\_WinIdNo Nein-Schaltfläche

gedrückt

Siehe Verwandte Befehle, WinDialogRun()

Dieser Befehl lädt das Fenster (alpha1) und zeigt es an. Ist kein Dialog-Objekt mit dem angegebenen Namen vorhanden oder ist die Berechtigung des Benutzers nicht ausreichend, wird kein Dialog angezeigt und der globale Fehlerwert auf rNoRec gesetzt.

Zur Darstellung des Fensters werden die Eigenschaften verwendet, die zum Entwurfszeitpunkt angegeben wurden. Kann aufgrund der Area-Eigenschaften das Fenster nicht im sichtbaren Bereich dargestellt werden, wird es automatisch in die Mitte des primären Bildschirms verschoben.

Je nach verwendetem Betriebssystem stehen unterschiedlich viele Ressourcen (Benutzer- und GDI-Objekte) zur Darstellung des Fensters zur Verfügung. Nach dem Laden des Fensters werden die verbleibenden Ressourcen überprüft. Stehen weniger als 10% zur Verfügung, wird der Dialog nicht geladen. Die Anweisung gibt 0 als Deskriptor zurück und der globale Fehlerwert wird auf ErrOutOfMemory gesetzt. Die zur Verfügung stehenden Benutzer- und GDI-Objekte können über den Info-Dialog und die Eigenschaften ObjectsUserLimit und ObjectsGDILimit ermittelt werden.



Weiterführende Informationen unter Limitationen des Anwendungsprozesses Folgende Optionen (int2) können angegeben werden:

- WinDialogAlwaysOnTop

Das Fenster (alpha1) wird immer im Vordergrund angezeigt.

- WinDialogApp

Der CONZEPT 16-Client wird versteckt.

- WinDialogCenter

Das Fenster (alpha1) wird zum Elternfenster (handle3) zentriert.

## Kontakt

- WinDialogCenterX

Das Fenster (alpha1) wird zum Elternfenster (handle3) horizontal zentriert.

- WinDialogCenterY

Das Fenster (alpha1) wird zum Elternfenster (handle3) vertikal zentriert.

- WinDialogCenterScreen

Das Fenster (alpha1) wird zum Bildschirm zentriert.

- WinDialogCenterScreenX

Das Fenster (alpha1) wird zum Bildschirm horizontal zentriert.

- WinDialogCenterScreenY

Das Fenster (alpha1) wird zum Bildschirm vertikal zentriert.

- WinDialogCreateHidden

Obsolet. Sollte nicht mehr verwendet werden.

- WinDialogMaximized

Das Fenster (alpha1) wird maximiert dargestellt.

- WinDialogMinimized

Das Fenster (alpha1) wird minimiert dargestellt.

- WinDialogNormal

Das Fenster (alpha1) wird weder minimiert noch maximiert dargestellt.

- WinDialogNoActivate

Das Fenster (alpha1) wird inaktiv gestartet.

Die Optionen WinDialogAlwaysOnTop, WinDialogApp und WinDialogNoActivate können mit einer der anderen Optionen kombiniert werden.

Wird (handle3) nicht angegeben, ist der CONZEPT 16-Client das Eltern-Objekt des Dialogs. Andernfalls ist das Objekt, dessen Objektdeskriptor in (handle3) angegeben ist, das Elternfenster. Als Elternfenster kann nur ein Fenster/MDI-Fenster-Objekt angegeben werden.

Erfolgt der Aufruf eines Fensters mit einem Elternfenster (int3), erscheint das Fenster nicht als separater Task in der Taskleiste.

Ein Wechsel des Fokus in das Elternfenster ist nicht möglich. Der Eingabefokus kann erst dann wieder in das Elternfenster zurückgelangen, wenn der Fenster geschlossen wurde.

Das Fenster wird mit dem Drücken einer Schaltfläche, deren Eigenschaft TypeButton auf WinBtnClose gesetzt ist, verlassen. Das Resultat ist der Wert der Eigenschaft ID der gedrückten Schaltfläche.

## Kontakt



**MDI-Fenster**-Objekte müssen mit dem Befehl WinAdd() bzw. WinAddByName() geladen werden.

### Beispiele:

```
// Fenster laden und anzeigenWinDialog('FrmKunden');if (ErrGet() != _ErrOk)...// Fenster 'Message
```

### Mögliche Laufzeitfehler:

ErrStringOverflow Fenstername (alpha1) länger als 40 Zeichen



## Kontakt

obj -> WinDialogBox(alpha1,  
alpha2, int3, int4, int5) : int  
Meldungsfenster aufrufen



obj	Elternfenster	
alpha1	Fenstertitel	
alpha2	Meldungstext	
	Fenstersymbol	
	<u>_WinIcoApplication</u>	Anwendungssymbol
	<u>_WinIcoInformation</u>	Informationssymbol
int3	<u>_WinIcoError</u>	Fehlersymbol
	<u>_WinIcoWarning</u>	Warnungssymbol
	<u>_WinIcoQuestion</u>	Fragesymbol
	Optionen	
	<u>_WinDialogOK</u>	OK-Schaltfläche
		anzeigen
	<u>_WinDialogOKCancel</u>	OK- und
		Abbrechen-Schaltfläche
		anzeigen
	<u>_WinDialogYesNo</u>	Ja- und
		Nein-Schaltfläche
int4		anzeigen
	<u>_WinDialogYesNoCancel</u>	Ja-, Nein- und
		Abbrechen-Schaltfläche
		anzeigen
	<u>_WinDialogAlwaysOnTop</u>	Meldungsfenster immer
		im Vordergrund
		darstellen
	<u>_WinDialogUtf8</u>	UTF-8-Zeichensatz
int5	Standard-Schaltflächennummer	
	Schaltflächen-ID	
	<u>_WinIdClose</u>	Schließen-Schaltfläche
		gedrückt
	<u>_WinIdOk</u>	OK-Schaltfläche gedrückt
Resultat	<u>_WinIdCancel</u>	Abbrechen-Schaltfläche
		gedrückt
	<u>_WinIdYes</u>	Ja-Schaltfläche gedrückt
	<u>_WinIdNo</u>	Nein-Schaltfläche gedrückt

Siehe Verwandte Befehle

Mit diesem Befehl wird ein Meldungsfenster aufgerufen. Bei Angabe eines Elternfensters in (obj) verhält sich das Meldungsfenster modal zu diesem Fenster.



Sofern das Meldungsfenster nicht modal aufgerufen werden soll, wird in (obj) 0 angegeben, und dieses vor den Befehlsargumenten aufgeführt:

```
WinDialogBox(0, alpha1, alpha2, int3, int4, int5);
```

Standardmäßig wird die Positionierung der DialogBox vom Betriebssystem

## Kontakt

übernommen (zentriert zum Bildschirm). Über die Eigenschaften DialogBoxTop und DialogBoxLeft des App-Objekts kann die Positionierung auch individuell vorgenommen werden.

In der Titelzeile wird der Text in (alpha1) ausgegeben. Der Meldungstext wird in (alpha2) übergeben. Dieser Text wird in einer Zeile innerhalb des Fensters angezeigt. Sind in der übergebenen Zeichenkette Zeilenumbrüche angegeben (StrChar(13)), können auch mehrzeilige Texte ausgegeben werden. Mit Tab (StrChar(9)) kann eine Formatierung der Ausgabe erfolgen. (int3) bestimmt das Symbol im Meldungsfenster. In (int4) werden die Schaltflächen definiert und in (int5) wird die Nummer der Standardschaltfläche angegeben. Die Schaltflächen werden von links (mit 1) nach rechts durchnummeriert. Sollen Zeichenketten im UTF-8-Zeichensatz ausgegeben werden, muss die Angabe der Schaltflächen in (int4) mit der Konstante WinDialogUtf8 kombiniert werden.

Die Option WinDialogAlwaysOnTop kann mit den Schaltflächen-Optionen kombiniert werden.

### Beispiele:

Der Befehl kann zur einfachen Fehlersuche im Prozedurtext verwendet werden. Es ist dabei zu beachten, dass durch den Aufruf einer Dialogbox der Eingabefokus in die Dialogbox wechselt und unter Umständen Ereignisse aufgerufen werden.

```
// Speichern-Abfrage durchführen if (WinDialogBox(0, 'Application', 'Save record?',
```

## Kontakt

**obj -> WinDialogResult([int1]) :**



**int**

**Fenster-Resultat setzen/ermitteln**

**obj**            **Fenster**  
                 **Neue Schaltflächen-ID**

**int1**

**(optional)**

**Resultat int Aktuelle Schaltflächen-ID**

**Siehe**            **Verwandte Befehle,**

**WinDialog(), WinDialogRun()**

**Dieser Befehl gibt die ID der gedrückten Schaltfläche zurück. Wenn (int1) übergeben wird, wird diese Zahl als ID gesetzt.**

**Beispiel:**

```
// Durchführung einer Schleife bis zum Drücken der Abbrechen-SchaltflächeWinDialog('Message', _Wi
```

**Mögliche Laufzeitfehler:**

**ErrHdlInvalid Fenster (obj) ungültig**

obj -> WinDialogRun([int1[,  
handle2]]) : int



Geladenes Fenster anzeigen

obj        Fenster  
          Optionen (optional;  
int1

      siehe Text)

**Elternfenster**  
handle2                    (optional)

      Resultat int Schaltflächen-ID

Verwandte Befehle,

Siehe        WinOpen(),  
          WinOpen(), Beispiel

Mit diesem Befehl wird ein geladenes Fenster-Objekt angezeigt. Das Objekt muss zuvor mit dem Befehl WinOpen() geladen worden sein. Als (obj) wird der zurückgegebene Deskriptor oder der Name des Fenster-Objektes mit einem vorangestellten \$ (zum Beispiel \$Meldung) verwendet.

Folgende Optionen (int1) können angegeben werden:

- WinDialogAlwaysOnTop

      Das Fenster (obj) wird immer im Vordergrund angezeigt.

- WinDialogApp

      Der CONZEPT 16-Client wird versteckt.

- WinDialogAsync

      Das Fenster (obj) wird asynchron zur laufenden Prozedur angezeigt.

- WinDialogCenter

      Das Fenster (obj) wird zum Elternfenster (handle2) zentriert.

- WinDialogCenterX

      Das Fenster (obj) wird zum Elternfenster (handle2) horizontal zentriert.

- WinDialogCenterY

      Das Fenster (obj) wird zum Elternfenster (handle2) vertikal zentriert.

- WinDialogCenterScreen

      Das Fenster (obj) wird zum Bildschirm zentriert.

- WinDialogCenterScreenX

      Das Fenster (obj) wird zum Bildschirm horizontal zentriert.

- WinDialogCenterScreenY

      Das Fenster (obj) wird zum Bildschirm vertikal zentriert.

- WinDialogCreateHidden

      Obsolet. Sollte nicht mehr verwendet werden.

- WinDialogMaximized

## Kontakt

Das Fenster (obj) wird maximiert dargestellt.

- WinDialogMinimized

Das Fenster (obj) wird minimiert dargestellt.

- WinDialogNormal

Das Fenster (obj) wird weder minimiert noch maximiert dargestellt.

- WinDialogNoActivate

Das Fenster (obj) wird inaktiv gestartet.

Die Optionen WinDialogAlwaysOnTop, WinDialogApp, WinDialogAsync und WinDialogNoActivate können mit einer der anderen Optionen kombiniert werden.

Wird (handle2) nicht angegeben, ist der CONZEPT 16-Client das Eltern-Objekt des Dialoges. Andernfalls ist das Objekt, dessen Objektdeskriptor in (handle2) angegeben ist, das Elternfenster. Als Elternfenster kann nur ein Fenster/MDI-Fenster-Objekt angegeben werden.

Der Wert der Eigenschaft ID der gedrückten Schaltfläche im Fenster wird als Resultat zurückgegeben.

Über die Funktion WinDialogResult() kann der Rückgabewert von WinDialogRun() individuell gesetzt werden.

Nach dem Schließen (entweder durch das Drücken einer Schließen-Schaltfläche oder durch die Ausführung der Anweisung WinClose()) befindet sich das Fenster noch im Speicher. Es kann also auch nach der Anweisung WinDialogRun() auf die Eigenschaften und Objekte des Dialoges zugegriffen werden. Der Dialog wird erst nach einem erneuten Aufruf von WinClose() aus dem Speicher entfernt und kann auch erst dann wieder geladen und angezeigt werden.


Beispiel:

```
// Fenster ladenWinOpen('Message');// Fenster zentriert anzeigenRes # $Message->WinDialogRun(_Wi
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Fenster (obj) ungültig

## Kontakt

WinOpen(alpha1[, int2[, int3]) : int System- 

Fenster/Fenster-Objekt laden

alpha1	<u>Fenstername</u>	
	<u>_WinComFileOpen</u>	System-Fenster
		"Datei öffnen" laden
	<u>_WinComFileSave</u>	System-Fenster
		"Datei speichern"
		laden
	<u>_WinComPath</u>	System-Fenster
		"Pfadauswahl" laden
	<u>_WinComPrint</u>	System-Fenster
		"Drucken" laden
	<u>_WinComPrintSetup</u>	System-Fenster
		"Druckeinrichtung"
		laden
	<u>_WinComFont</u>	System-Fenster
int2		"Schriftart" laden
	<u>_WinComOdbc</u>	ODBC-Auswahl-Dialog
		laden
	<u>_WinC16Info</u>	CONZEPT 16-Fenster
		"Info" laden
	Optionen (optional)	
	<u>_WinOpenDialog</u>	Fenster-Objekt laden
	<u>_WinOpenLock</u>	Fenster-Objekt laden
		und sperren
	<u>_WinOpenEventsOff</u>	Fenster-Objekt ohne Ereignis-Verarbeitung
int3		laden
	<u>_WinOpenUnicode</u>	Öffnen- ( <u>_WinComFileOpen</u> ) oder Speichern-Dialog ( <u>_WinComFileSave</u> ) im Unicode-Unterstützung öffnen

Druckerausgabe-Objekt (optional)

Resultat handle Fenster-Objekt 

Siehe Verwandte Befehle, WinDialogRun(),

WinClose(), WinSave()

Mit diesem Befehl wird ein System-Fenster oder ein Fenster-Objekt in den Hauptspeicher geladen, aber noch nicht angezeigt. Mit dem Befehl WinDialogRun() erfolgt die Anzeige auf dem Bildschirm.

Die Funktion gibt den Deskriptor des System-Fenster/Fenster-Objekts zurück. Konnte das Fenster nicht geladen werden oder besitzt der angemeldete Benutzer keine ausreichende Berechtigung, ist das Ergebnis 0 und der globale Fehlerwert ist auf rNoRec gesetzt.

## Kontakt

Je nach verwendetem Betriebssystem stehen unterschiedlich viele Ressourcen (Benutzer- und GDI-Objekte) zur Darstellung des Fensters zur Verfügung. Nach dem Laden des Fensters werden die verbleibenden Ressourcen überprüft. Stehen weniger als 10% zur Verfügung, wird der Dialog nicht geladen. Die Anweisung gibt 0 als Deskriptor zurück und der globale Fehlerwert wird auf ErrOutOfMemory gesetzt. Die zur Verfügung stehenden Benutzer- und GDI-Objekte können über den Info-Dialog und die Funktionen ObjectsUserLimit und ObjectsGDILimit ermittelt werden.



Weiterführende Informationen unter Limitationen des Anwendungsprozesses Laden eines System-Fensters

Je nach Art des System-Fensters wird in (alpha1) eine der Stringkonstanten übergeben. (int2) wird nicht angegeben. Bei den Dialogen "Drucken" und "Druckeinrichtung" muss in (obj3) der Deskriptor eines Druckerausgabe-Objektes übergeben werden.

Folgende System-Fenster (int1) stehen zur Verfügung:

- WinComFileOpen

Ein Fenster zum Öffnen einer Datei wird geladen.

- WinComFileSave

Ein Fenster zum Speichern einer Datei wird geladen.

- WinComPath

Ein Fenster zum Auswählen eines Pfades wird geladen.

- WinComPrint

Ein Fenster zum Drucken wird geladen.

- WinComPrintSetup

Ein Fenster zum Anpassen der Drucker-Einstellungen wird geladen.

- WinComFont

Ein Fenster zum Auswählen der Schriftart wird geladen.

- WinComOdbc

Ein Dialog zur Angabe einer ODBC-Datenquelle wird geladen.

- WinC16Info

Der CONZEPT 16-Info-Dialog wird geladen.

Die System-Fenster geben den Wert 0 zurück, wenn die OK-Schaltfläche gedrückt wurde. Wird das Fenster geschlossen, wird -1 zurückgegeben.

Die Eigenschaften, die bei System-Fenstern gesetzt/ermittelt werden können, sind bei den entsprechenden Konstanten erläutert.

Laden eines Fenster-Objekts

## Kontakt

Der Name des Fenster-Objektes wird in (alpha1) und in (int2) die Option \_WinOpenDialog angegeben. Nach dem Öffnen eines Objekts ist der Suchpfad automatisch auf dieses Objekt gesetzt. Es werden also keine Namen von zuvor geladenen Objekten gefunden. Der Suchpfad kann mit dem Befehl WinSearchPath() gesetzt werden.

### Beispiel 1:

```
// Fenster-Objekt laden und anzeigen// Fenster 'Message' in tHdl ladentHdl # WinOpen('Message', _
```

### Beispiel 2:

```
// System-Fenster zum Speichern einer Datei aufrufen// System-Fenster "Datei speichern" in tHdl l
```

### Beispiel 3:

```
// System-Fenster "Druckeinrichtung" laden und aufrufen// Druckausgabe-Objekt ladentPrintDev # Pr
```

## Laden eines Fensters zum Bearbeiten

Wird ein Fenster-Objekt mit der Anweisung WinOpen(..., \_WinOpenDialog | \_WinOpenLock) geöffnet, kann es verändert und wieder in der Datenbank gespeichert werden. Die Speicherung erfolgt mit der Anweisung WinSave(). Das Objekt kann nur dann geladen und gesperrt werden, wenn es zu diesem Zeitpunkt nicht durch den eigenen oder einen anderen Benutzer geändert wird. Die Sperre bleibt bis zum Schließen des Objekts mit der Anweisung WinClose() erhalten. Wird in dieser Zeit der Dialog im Designer geöffnet erscheint die Meldung "Frame mit Name ... wird bereits editiert."

Ist das Fenster bereits zum Ändern geöffnet, wird von der WinOpen()-Anweisung 0 zurückgegeben und der globale Fehlerwert auf ErrLocked gesetzt. Besitzt der Benutzer keine Rechte zum Ändern des Fensters, ist der Fehlerwert ErrRights.

Beim Öffnen zum Ändern wird das Objekt geladen, ohne das Ereignisse ausgeführt werden. Das Objekt kann anschließend auch mit den Anweisungen WinDialogRun() bzw. WinAdd() angezeigt werden, ohne das Ereignisse aufgerufen werden.

## Laden eines Fensters ohne Ereignis-Verarbeitung

Wird ein Fenster-Objekt mit der Anweisung WinOpen(..., \_WinOpenDialog | \_WinOpenEventsOff) geöffnet, ist die Ereignisverarbeitung des Dialogs deaktiviert. In allen anderen Dialogen werden die Ereignisse weiterhin ausgeführt.

## Laden eines Systemfensters mit Unicode-Unterstützung

Wird ein Öffnen- ( \_WinComFileOpen) bzw. Speichern-Dialog ( \_WinComFileSave) mit der Option \_WinOpenUnicode gestartet, werden die Eigenschaften Caption, FileName, PathName, FileNameExt und FileFilter als UTF8-kodierte Zeichenketten erwartet und zurückgegeben. Zur Umwandlung zwischen UTF8 und CONZEPT 16-Zeichensatz können folgende Makros verwendet werden:

```
define{ // Konvertierung von Zeichen: CONZEPT 16-Zeichensatz UTF8 mC16toUtf8(aText) : StrC
```



## Kontakt

**Mögliche Laufzeitfehler:**

**ErrStringOverflow Fenstername (alpha1) länger als 40 Zeichen**

## Kontakt

obj -> WinSave(int1[,  
alpha2]) : int; Fenster-Objekt  
speichern



obj      Deskriptor des Fenster-Objekts  
         Optionen  
          \_WinSaveDefault bestehendes  
                                 Objekt nicht  
int1                                   überschreiben  
          \_WinSaveOverwrite bestehendes  
                                 Objekt  
                                 überschreiben

alpha2    Name des Fenster-Objekts  
          (optional)

Resultat int      Fehlerwert

Siehe      WinOpen(), WinCreate()

Mit dieser Anweisung kann ein mit WinOpen() geöffnetes oder mit WinCreate() erstelltes Fenster-Objekt in der Datenbank gespeichert werden. Wurde das Objekt mit WinOpen() geöffnet, muss dies mit der Option WinOpenLock gesperrt worden sein.

Als (obj) wird der Deskriptor des Fenster-Objekts übergeben, der von WinOpen() bzw. WinCreate() zurückgegeben wurde. Als Optionen stehen folgende Konstanten zur Verfügung:

- \_WinSaveDefault (0) - bestehendes Objekt nicht überschreiben

Das Objekt wird unter dem in (alpha2) übergebenem Namen gespeichert. Existiert das Objekt bereits, wird der Fehler \_ErrExists zurückgegeben.

- \_WinSaveOverwrite (1) - bestehendes Objekt überschreiben

Das Objekt wird unter dem in (alpha2) angegebenen Namen gespeichert, auch dann, wenn ein gleichnamiges Objekt bereits existiert. Das entsprechende Objekt wird überschrieben.

Wird kein Name angegeben, wird der Name des Objekts (Eigenschaft Name) verwendet. Es wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Der Rückgabewert der Anweisung kann mit folgenden Konstanten verglichen werden:

<u>_ErrOk</u>	Das Objekt wurde gespeichert (kein Fehler).
<u>_ErrNameInvalid</u>	Der Name für das Objekt ist ungültig (Argument (alpha2)).
<u>_ErrLocked</u>	Das Objekt wird vom aktuellen Client im Designer editiert oder ist bereits durch <u>_WinOpenLock</u> geöffnet.
<u>_ErrExists</u>	Ein Objekt unter diesem Namen existiert bereits in der Datenbank und <u>_WinSaveOverwrite</u> ist nicht angegeben.
<u>_ErrRights</u>	Benutzerrechte nicht ausreichend.
<u>_rDeadlock</u>	Verklemmung aufgetreten

Mögliche Laufzeitfehler:

## Kontakt

### ErrHdlInvalid

Der in (obj) übergebene Deskriptor ist ungültig, ist kein Fenster-Objekt oder nicht gesperrt.

### ErrValueInvalid

In (int1) ist ein ungültiger Wert übergeben worden.

### ErrStringOverflow

In (alpha2) wurde ein Name mit mehr als 40 Zeichen übergeben.

## Kontakt

obj -> WinUrmDialog(int1)  Fenster der Benutzerverwaltung  
laden und ausführen

obj Elternfenster

Optionen

int1 \_WinUrmFlagHelpTipOnSysProps Anzeige der  
Eigenschaftsnamen  
im Helptipp

Siehe Verwandte Befehle, WinDialog(), WinDialogRun()

Mit dieser Anweisung wird die CONZEPT 16-Benutzerverwaltung aufgerufen. Sie entspricht der Benutzerpflege, die in der Entwicklungsumgebung aufgerufen werden kann. Der Dialog ist in verschiedenen Sprachen verfügbar. Die Sprache kann über die Eigenschaft LangDisplay des App-Objekts gesteuert werden.

Als Objekt kann ein Fenster-Objekt übergeben werden. Das Fenster ist dann das Eltern-Objekt zum Benutzerpflege-Dialog. Wird kein Objekt angegeben, muss der Benutzerpflege-Dialog wie folgt aufgerufen werden:

```
WinUrmDialog(0, <Optionen>);
```

In (int1) kann die Option \_WinUrmFlagHelpTipOnSysProps angegeben werden. Sie bewirkt, dass bei Einstellungen, die einer Eigenschaft entsprechen, der Name der Eigenschaft im Helptip angezeigt wird.

Beispiel:

```
tHdlFrame->WinUrmDialog(_WinUrmFlagHelpTipOnSysProps);
```

Konstanten für Dialog-Befehle

Konstanten für Dialog-Befehle

Siehe Dialog-Befehle

Konstanten

- WinAddHidden
- WinDialogAlwaysOnTop
- WinDialogApp
- WinDialogAsync
- WinDialogBoxDefault
- WinDialogBoxUseFont
- WinDialogBoxUseFontButton
- WinDialogBoxUseTextButton
- WinDialogCenter
- WinDialogCenterScreen
- WinDialogCenterScreenX
- WinDialogCenterScreenY
- WinDialogCenterX
- WinDialogCenterY
- WinDialogCreateHidden
- WinDialogMaximized
- WinDialogMinimized
- WinDialogNoActivate
- WinDialogOK
- WinDialogOKCancel
- WinDialogUtf8
- WinDialogYesNo
- WinDialogYesNoCancel
- WinIcoApplication
- WinIcoError
- WinIcoExternFile
- WinIcoInformation
- WinIcoQuestion
- WinIcoWarning
- WinOpenDialog
- WinOpenEventsOff
- WinOpenLock
- WinOpenUnicode
- WinUpdScrollPos
- WinUrmFlagHelpTipOnSysProps

**\_WinAddHidden**

**MDI-Fenster unsichtbar laden**

**Wert 1 / 0x00000001**

**Verwandte**

Siehe Befehle, WinAdd(),

WinAddByName()

Option bei WinAdd() und WinAddByName() durch die ein MDI-Fenster unsichtbar geladen werden kann.

Nach dem Laden des MDI-Fensters stehen alle Eigenschaften und Objekte des Fensters zur Verfügung. Eventuell definierte Ereignisse werden verarbeitet. Mit WinUpdate() und dem Parameter \_WinUpdOn wird das Fenster gezeichnet.

## Kontakt

**\_WinCopyDefault**

**Objekte mit allen Eigenschaften und Ereignissen kopieren**

Wert **0/**

**0x00000000**

**Verwandte**

Siehe **Befehle**,

**WinCopy()**

Option bei **WinCopy()** durch die alle Eigenschaften und Ereignisse des Objektes und aller Unterobjekte kopiert werden.

**\_WinCopyNoEvents**

**Objekte ohne Ereignisse kopieren**

Wert <sup>1/</sup>

**0x00000001**

**Verwandte**

Siehe **Befehle**,

**WinCopy()**

Option bei **WinCopy()** durch die keine Ereignisse des Objektes und aller Unterobjekte kopiert werden.



**\_WinDialogAlwaysOnTop**

**Fenster ist immer im Vordergrund**

Wert **-2.147.483.648 /**

**0x80000000**

**Verwandte**

**Befehle,**

Siehe **WinDialog(),**

**WinDialogRun(),**

**WinDialogBox()**

Option bei **WinDialog(), WinDialogRun() und WinDialogBox().**

Wird diese Konstante angegeben, wird das Fenster immer im Vordergrund angezeigt. Die Option kann nicht bei Objekten **AppFrame, TrayFrame** und den **Systemdialogen** verwendet werden.

## Kontakt

**\_WinDialogApp**

**CONZEPT 16-Client unsichtbar setzen**

Wert 536.870.912 /

**0x20000000**

**Verwandte**

Siehe Befehle,

WinDialog(),

WinDialogRun()

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der CONZEPT 16-Client nicht mehr auf dem Desktop und in der Taskbar angezeigt.

## Kontakt

**\_WinDialogAsync**

Fenster wird asynchron zur laufenden Prozedur aufgerufen

Wert 268.435.456 /

**0x10000000**

Verwandte

Siehe Befehle,

WinDialogRun()

Ist diese Konstante bei WinDialogRun() angegeben, wird die Prozedur nach dem Aufrufen des Dialoges fortgesetzt.



Es darf immer nur ein asynchroner Dialog gestartet sein. Wird ein weiterer asynchroner Dialog gestartet, wird der Laufzeitfehler ErrValueInvalid erzeugt.

In der Regel erfolgt im weiteren Programmverlauf eine Schleife, die in regelmäßigen Abständen überprüft, ob der asynchrone Dialog geschlossen wurde (siehe WinDialogResult()).

Damit die Prozessorzeit nicht (fast) ausschließlich zur Verarbeitung dieser Schleife verwendet wird, kann mit dem Befehl WinSleep() eine Wartezeit implementiert werden, die von CONZEPT 16 genutzt wird, um die Windows-Nachrichten-Schleife abzufragen und ein eingetroffenes Ereignis zu verarbeiten. Die Dauer der Wartezeit sollte von der Verarbeitung innerhalb der Schleife abhängig gemacht werden. Es kann ebenfalls eine Dauer von 0 Millisekunden angegeben werden.

**Beispiel:**

```
// Frame ladentFrame # WinOpen('Frame', _WinOpenDialog); // Frame asynchron startentFrame->WinDial
```



Damit das Fenster über die Schaltfläche geschlossen werden kann, muss die Eigenschaft ID auf den Wert gesetzt werden, der in der Schleife überprüft wird. In diesem Fall \_WinIdClose, es kann aber auch jede andere ID verwendet werden.

Werden in dem Dialog vordefinierte Schaltflächen verwendet (siehe TypeButton), werden automatisch den Schaltflächen entsprechende IDs gesetzt (siehe ID-Ausprägungen).

Um zu verhindern, dass der asynchrone Dialog mit  geschlossen werden kann, muss die Eigenschaft StyleCloseBox auf false gesetzt werden. Prozedural kann das Fenster über die Anweisung tFrame->WinDialogResult(\_WinIdClose) geschlossen werden.

## Kontakt

**\_WinDialogBoxDefault**

**Standard-Einstellung für Nachrichtenfenster.**

**Wert 0 / 0x00000000**

**Siehe WinDialogBox()**

**Option bei der Eigenschaft DialogBoxFlags des \_App-Objektes.**

**Wird dieser Wert angegeben, wird in Nachrichtenfenstern, die mit WinDialogBox() angezeigt werden, die Standard-Schriftart und die Standard-Schaltflächenbeschriftung verwendet.**

## Kontakt

**\_WinDialogBoxUseFont**

Schriftart für Nachrichtenfenster.

Wert 1 / 0x00000001

Siehe **WinDialogBox()**

Option bei der Eigenschaft **DialogBoxFlags** des **\_App**-Objektes.

Nachfolgende Aufrufe von **WinDialogBox()** verwenden für die Textanzeige die

Schriftart, die in der Eigenschaft **DialogBoxFont** hinterlegt ist. Die Schriftart wird auch für die Anzeige der Schaltflächen verwendet, sofern die Option

**\_WinDialogBoxUseFontButton** nicht gesetzt ist. Die Größe des Nachrichtenfensters wird entsprechend der Schriftart angepasst.



Diese Option wirkt sich nicht aus, wenn das Elternfenster der Dialogbox im **Modern Theme Style** dargestellt (Eigenschaft **StyleTheme** = **\_WinStyleThemeModern**) wird.

## Kontakt

**\_WinDialogBoxUseFontButton**

Schriftart für Schaltflächen in Nachrichtenfönstern.

Wert 2 / 0x00000002

Siehe WinDialogBox()

Option bei der Eigenschaft DialogBoxFlags des \_App-Objektes.

Nachfolgende Aufrufe von WinDialogBox() verwenden für die Textanzeige die Schriftart, die in der Eigenschaft DialogBoxFontButton hinterlegt ist. Die Größe der Schaltflächen wird entsprechend der Schriftgröße angepasst - ebenso die daraus resultierende Größe des Nachrichtenfönsters.



Diese Option wirkt sich nicht aus, wenn das Elternfenster der Dialogbox im Modern Theme Style dargestellt (Eigenschaft StyleTheme = \_WinStyleThemeModern) wird.

## Kontakt

**\_WinDialogBoxUseTextButton**

**Beschriftung für Schaltflächen in Nachrichtenfestern.**

**Wert 4 / 0x00000004**

**Siehe WinDialogBox()**

**Option bei der Eigenschaft DialogBoxFlags des \_App-Objektes.**

**Nachfolgende Aufrufe von WinDialogBox() verwenden für die Textanzeige der Schaltflächen die Beschriftung, die in der Eigenschaft DialogBoxTextButton hinterlegt ist.**

## Kontakt

**\_WinDialogCenter**

Fenster wird zum Elternobjekt zentriert

Wert 3 / 0x00000003

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenterScreen

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog zum Elternobjekt zentriert.

Wird die Konstante bei der Eigenschaft DialogBoxArrange des \_App-Objektes angegeben, dann werden die folgenden mit WinDialogBox() erzeugten Dialog-Boxen zum Elternobjekt zentriert.

Damit sich die Eigenschaft DialogBoxArrange auswirkt, müssen die Eigenschaften DialogBoxLeft und DialogBoxTop auf den Wert -1 gesetzt sein.

Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es auf dem nächsten Bildschirm zentriert dargestellt.



## Kontakt

**\_WinDialogCenterScreen**

Fenster wird zum Bildschirm zentriert

Wert 12 / 0x0000000C

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenter

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog zum Bildschirm zentriert.

Wird die Konstante bei der Eigenschaft DialogBoxArrange des \_App-Objektes angegeben, dann werden die folgenden mit WinDialogBox() erzeugten Dialog-Boxen zum Bildschirm zentriert. Damit sich die Eigenschaft DialogBoxArrange auswirkt, müssen die Eigenschaften DialogBoxLeft und DialogBoxTop auf den Wert -1 gesetzt sein.

Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

## Kontakt

**\_WinDialogCenterScreenX**

Fenster wird horizontal zum Bildschirm zentriert Wert 4 /  
0x00000004

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenterX

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog horizontal zum Bildschirm zentriert.

Wird die Konstante bei der Eigenschaft DialogBoxArrange des \_App-Objektes angegeben, dann werden die folgenden mit WinDialogBox() erzeugten Dialog-Boxen horizontal zum Bildschirm zentriert. Damit sich die Eigenschaft DialogBoxArrange auswirkt, müssen die Eigenschaften DialogBoxLeft und DialogBoxTop auf den Wert -1 gesetzt sein.

Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

## Kontakt

**\_WinDialogCenterScreenY**

Fenster wird vertikal zum Bildschirm zentriert Wert 8  
/ 0x00000008

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenterY

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog vertikal zum Bildschirm zentriert.

Wird die Konstante bei der Eigenschaft DialogBoxArrange des \_App-Objektes angegeben, dann werden die folgenden mit WinDialogBox() erzeugten Dialog-Boxen vertikal zum Bildschirm zentriert. Damit sich die Eigenschaft DialogBoxArrange auswirkt, müssen die Eigenschaften DialogBoxLeft und DialogBoxTop auf den Wert -1 gesetzt sein.

Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

## Kontakt

**\_WinDialogCenterX**

Fenster wird horizontal zum Elternobjekt zentriert Wert 1 /  
0x00000001

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenterY

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog horizontal zum Elternobjekt zentriert.

Wird die Konstante bei der Eigenschaft DialogBoxArrange des \_App-Objektes angegeben, dann werden die folgenden mit WinDialogBox() erzeugten Dialog-Boxen horizontal zum Elternobjekt zentriert. Damit sich die Eigenschaft DialogBoxArrange auswirkt, müssen die Eigenschaften DialogBoxLeft und DialogBoxTop auf den Wert -1 gesetzt sein.

Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

## Kontakt

**\_WinDialogCenterY**

Fenster wird vertikal zum Elternobjekt zentriert Wert 2  
/ 0x00000002

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenterX

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog vertikal zum Elternobjekt zentriert.

Wird die Konstante bei der Eigenschaft DialogBoxArrange des \_App-Objektes angegeben, dann werden die folgenden mit WinDialogBox() erzeugten Dialog-Boxen vertikal zum Elternobjekt zentriert. Damit sich die Eigenschaft DialogBoxArrange auswirkt, müssen die Eigenschaften DialogBoxLeft und DialogBoxTop auf den Wert -1 gesetzt sein.

Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

**\_WinDialogCreateHidden**

**Fenster erst nach Erzeugung anzeigen**

Wert 1.073.741.824 /

**0x40000000**

**Verwandte**

Siehe Befehle,

WinDialog(),

WinDialogRun()



Die Konstante ist obsolet und sollte nicht mehr verwendet werden.

**\_WinDialogMaximized**

Fenster wird maximiert dargestellt

Wert 16 / 0x00000010

Verwandte

Befehle,

WinDialog(),

Siehe WinDialogRun(),

WinAdd(),

WinAddByName(),

WinUpdate(),

WinInfo()

Je nach Befehl hat **\_WinDialogMaximized** folgende Bedeutung:

- WinDialog(), WinDialogRun(), WinAdd() und WinAddByName() Dialog wird maximiert geladen.
- WinUpdate()  
Dialog wird zur Laufzeit maximiert. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.
- WinInfo()  
Liefert WinInfo( WinState) den Wert **\_WinDialogMaximized** zurück, ist der Dialog maximiert.

**\_WinDialogMinimized**

Fenster wird minimiert dargestellt

Wert 32 / 0x00000020

Verwandte

Befehle,

WinDialog(),

Siehe WinDialogRun(),

WinAdd(),

WinAddByName(),

WinUpdate(),

WinInfo()

Je nach Befehl hat **\_WinDialogMinimized** folgende Bedeutung:

- WinDialog(), WinDialogRun(), WinAdd() und WinAddByName() Dialog wird minimiert geladen.
- WinUpdate()  
Dialog wird zur Laufzeit minimiert. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.
- WinInfo()  
Liefert WinInfo(\_WinState) den Wert **\_WinDialogMinimized** zurück ist der Dialog minimiert.



**\_WinDialogNoActivate**

Fenster wird inaktiv gestartet

Wert 16.777.216 / 0x01000000

**Verwandte Befehle,**

Siehe WinDialog(),

WinDialogRun(),

WinDialogCenterScreen

Ist diese Konstante bei WinDialog() bzw. WinDialogRun() angegeben, wird der Dialog inaktiv gestartet, das heißt er bekommt solange nicht den Fokus, bis der Benutzer ihn anwählt oder die Aktivierung prozedural durchgeführt wird (Beispielweise über WinFocusSet()).

Die Option kann nicht bei Objekten AppFrame und TrayFrame verwendet werden.

## Kontakt

**\_WinDialogOk**

Fenster hat eine OK-Schaltfläche

Wert 0

Verwandte

Siehe Befehle,

WinDialogBox()

Wird diese Konstante bei dem Befehl WinDialogBox() angegeben, besitzt das Fenster eine OK-Schaltfläche.

## Kontakt

**\_WinDialogOkCancel**

Fenster hat eine OK- und eine Abbrechen-Schaltfläche Wert 2

### **Verwandte**

Siehe **Befehle**,

**WinDialogBox()**

Wird diese Konstante bei dem Befehl **WinDialogBox()** angegeben, besitzt das Fenster eine OK- und eine Abbrechen-Schaltfläche.

**\_WinDialogUtf8**

**Text in UTF-8 darstellen**

**Wert 268.435.456 /**

**0x10000000**

**Verwandte**

**Siehe Befehle,**

**WinDialogBox()**

**Wird diese Konstante bei dem Befehl WinDialogBox() angegeben, können der Dialog-Box Zeichenketten im UTF-8-Zeichensatz übergeben werden.**

## Kontakt

**\_WinDialogYesNo**

Fenster hat eine Ja- und eine Nein-Schaltfläche Wert 3

### **Verwandte**

Siehe **Befehle**,

### **WinDialogBox()**

Wird diese Konstante bei dem Befehl **WinDialogBox()** angegeben, besitzt das Fenster eine Ja- und eine Nein-Schaltfläche.

## Kontakt

**\_WinDialogYesNoCancel**

Fenster hat eine Ja-, eine Nein und eine Abbrechen-Schaltfläche Wert 4

### **Verwandte**

Siehe **Befehle**,

### **WinDialogBox()**

Wird diese Konstante bei dem Befehl **WinDialogBox()** angegeben, besitzt das Fenster eine Ja-, eine Nein- und eine Abbrechen-Schaltfläche.

**\_WinOpenDialog**

**Als Dialog öffnen**

**Wert 1 / 0x01**

**Siehe WinOpen()**

**Option bei WinOpen(). Lädt das Fenster als Dialog.**

## Kontakt

**\_WinUrmFlagHelpTipOnSysProps**

**Anzeige der Eigenschaftsnamen im Helptip**

**Wert 1**

**Siehe WinUrmDialog()**

**Option bei WinUrmDialog(). Ist diese Option angegeben werden bei Einstellungen, die eine Eigenschaft repräsentieren, der Name der Eigenschaften im Helptip angezeigt.**



### GanttGraph-Befehle

Befehle zum Bearbeiten eines GanttGraph-Objekts

Verwandte  
Befehle,

Siehe GanttGraph,  
Befehlsgruppen,  
Befehlsliste

Befehle

- WinGanttBoxAdd
- WinGanttCellInfo
- WinGanttIvlAdd
- WinGanttIvlRemove
- WinGanttLineAdd

## Kontakt

obj -> WinGanttIvlAdd(int1, int2, int3[,  
alpha4[, alpha5]]) : handle Intervall zu  
GanttGraph hinzufügen



obj        GanttGraph  
int1       Horizontale Position  
int2       Vertikale Position  
int3       Intervalllänge  
            Intervallname  
  
alpha4     (optional)  
alpha5     Intervalltitel (optional)  
Resultat handle Intervall-Objekt  
            Verwandte Befehle,  
Siehe      Interval, GanttGraph,  
            WinGanttIvlRemove()

Dieser Befehl fügt einem GanttGraph-Objekt (obj) ein Interval-Objekt hinzu. Die Koordinaten werden ab Zelle (0,0) gezählt. Falls ein Intervall hinzugefügt wird, das außerhalb des durch CellCountHorz und CellCountVert definierten Bereichs liegt, wird CellCountHorz bzw. CellCountVert entsprechend erweitert. Optional können Name (siehe Name) und Titel (siehe Caption) übergeben werden.

Im erfolgreichen Fall ist das Resultat ein Deskriptor auf das Interval-Objekt. Im Fehlerfall ist das Ergebnis 0.

Die Hintergrundfarbe der Interval-Objekte kann Transparenzen enthalten. Ein Farbwert mit Transparenzen wird mit dem Befehl WinColorOpacitySet() erzeugt.

Beispiel:

```
// Intervall in der 1. Spalte und der 3. Zeile, mit einer Länge// von 4, dem Namen 'ivl1' und dem
```

Mögliche Laufzeitfehler:

ErrHdlInvalid GanttGraph (obj) ungültig

## Kontakt

```
obj -> WinGanttCellInfo(point1, var point2) : logic
```

## Fensterkoordinaten in Zellkoordinaten umrechnen

**obj**                      **GanttGraph**

### Fensterkoordinaten relativ zur linken oberen

**point1**      **Objektecke**

**var point2 Zellkoordinaten**

**Resultat** logic Fensterkoordinaten liegen auf einer Zelle

## Verwandte Befehle, GanttGraph,

**Siehe**

## PrtGanttGraph

Mit dem Befehl kann im GanttGraph- und im PrtGanttGraph ermittelt werden, welche Zellkoordinaten an einer bestimmten Koordinate (point1) relativ zur linken oberen Objektecke liegen. Die Zellkoordinaten werden nach erfolgreicher Durchführung in (point2) geschrieben. Die Zellkoordinaten beginnen an der Position 0 / 0.

**Befindet sich an den Fensterkoordinaten eine Zelle, ist das Resultat true, ansonsten false.**

### Beispiel:

```
sub EvtMouseEvent( aEvt      : event;    // Ereignis aButton      : int;
```

### Mögliche Laufzeitfehler:

**ErrHdlInvalid** Das Objekt (obj) ist weder ein GanttGraph- noch ein PrtGanttGraph-Objekt.

## Kontakt



**obj -> WinGanttLineAdd(int1[, int2,  
logic3, alpha4]) : handle**

**Hilfslinie zu GanttGraph hinzufügen**

**obj**      GanttGraph

**int1**      Linienanfangspunkt

**int2**      Linienfarbe (optional)

Linienausrichtung

(optional)

**logic3**    true Horizontal

false Vertikal

**alpha4**    Liniennamen

**Resultat** handle Linien-Objekt

Verwandte Befehl,

**Siehe**      Line, GanttGraph,

WinGanttIvlRemove()

Dieser Befehl fügt einem GanttGraph-Objekt (obj) ein Line-Objekt hinzu. Der Parameter (int1) gibt die Nummer der Zeile/Splte an, ab der die Hilfslinie gezeichnet wird. Falls (logic3) fehlt oder den Wert true hat, wird eine horizontale Hilfslinie gezeichnet, ansonsten eine vertikale. Fehlt (int2) wird WinColHighLight als Linienfarbe benutzt.

Im erfolgreichen Fall ist das Resultat ein Deskriptor auf das Line-Objekt. Im Fehlerfall ist das Ergebnis 0.

**Beispiel:**

```
// Hilfslinie in der 2. Spalte in hellrot und mit vertikaler Ausrichtung erzeugen$GanttGraph->Win
```

**Mögliche Laufzeitfehler:**

ErrHdlInvalid GanttGraph (obj) ungültig

## Kontakt

obj -> WinGanttBoxAdd(rect1[,  
int2[, alpha3[, alpha4]) :  
handle



Box zu GanttGraph hinzufügen

obj        GanttGraph  
rect1      Box-Koordinaten  
            Box-Füllfarbe  
int2  
            (optional)

alpha3    Name der Box  
            (optional)

alpha4    Caption der Box  
            (optional)

Resultat handle Box-Objekt

Verwandte Befehle,

Siehe      WinGanttIvlRemove(),  
            GanttGraph, Box

Dieser Befehl fügt einem GanttGraph-Objekt (obj) ein Box-Objekt hinzu. In (rect1) wird die linke obere und rechte untere Ecke des Rechtecks übergeben. Die Angabe erfolgt in Zellkoordinaten. Fehlt (int2), wird WinColHighLight als Füllfarbe verwendet.

Über die Parameter (alpha3) und (alpha4) kann der Name und die Caption des Box-Objekts angegeben werden. Werden die Parameter nicht übergeben, bleiben die Eigenschaften leer.



Damit die Caption (alpha4) angezeigt wird, muss beim GanttGraph in der Eigenschaft GanttFlags der Wert WinGanttIvlBoxShowText gesetzt sein.

Im erfolgreichen Fall ist das Resultat ein Deskriptor auf das Box-Objekt. Im Fehlerfall ist das Ergebnis 0.

Die Hintergrundfarbe der Box-Objekte kann Transparenzen enthalten. Ein Farbwert mit Transparenzen wird mit dem Befehl WinColorOpacitySet() erzeugt.

Beispiel:

```
// Box in erste Zeile und erster Spalte mit drei Zellen Breite// und 2 Zellen Höhe und schwarzer
```

Mögliche Laufzeitfehler:

ErrHdlInvalid GanttGraph (obj) ungültig

## Kontakt

obj -> WinGanttIvlRemove()  Box/Intervall/Linie aus  
GanttGraph entfernen

objBox/Interval/Line

### Verwandte

Siehe Befehl, Box,

Interval, Line,

GanttGraph

Dieser Befehl entfernt ein Box-/Interval-/Line-Objekt (obj) aus einem  
GanttGraph-Objekt.

Beispiel:

```
// Alle Box-, Intervall- oder Linien-Objekte aus einem GanttGraph entfernen  
sub ObjectsRemove ( ag
```

Mögliche Laufzeitfehler:

ErrHdlInvalid GanttGraph (obj) ungültig

## Menu-Befehle

Befehle zum Bearbeiten eines Menu-Objekts

Verwandte  
Befehle,

Siehe MenuItem,  
Befehlsgruppen,  
Befehlsliste

Befehle

- WinMenuContext
- WinMenuItemAdd
- WinMenuItemRemove

## Kontakt

obj -> WinMenuItemAdd([alpha1[,  
alpha2[, int3]]) : handle Menüeintrag zu  
Menü hinzufügen



obj        Menü  
alpha1    Menüeintragsname (optional)  
alpha2    Menüeintragstitel (optional)  
int3       Menüeintragsposition (optional)  
            Deskriptor des neuen

### Resultat handle Menüeintrags

Siehe       Verwandte Befehle

Dieser Befehl fügt einen Menüeintrag zu einem Menü-Objekt (obj) hinzu. Wird in (obj) ein Menü oder ein Kontextmenüs übergeben, wird der Menüeintrag in der ersten Ebene angelegt. Handelt es sich um ein Menü, wird der Eintrag in der Menüzeile angelegt.



Bei einem Kontextmenü muss das Anlegen eines Menüeintrags in dem Ereignis EvtMenuInitPopup vorgenommen werden.

Bei der Übergabe eines bestehenden Menüeintrags in (obj) wird in dessen Untermenü der neue Eintrag hinzugefügt. Existiert noch kein Untermenü, wird ein neues angelegt. In (int3) wird die Position des Menüeintrags angegeben. Wird hier 0 angegeben oder kein Parameter angegeben, wird der Eintrag immer am Ende hinzugefügt. Name (alpha1) (siehe Name) und Titel (alpha2) (siehe Caption) können direkt beim Aufruf bestimmt werden, alle weiteren Eigenschaften müssen mit Hilfe des Resultats gesetzt werden.

Als Resultat wird der Deskriptor des neuen Menüeintrags zurückgeliefert. Im Fehlerfall ist das Ergebnis 0.

Beispiel:

```
// Menüeintrag 'Kopieren' zu Menü tMenuItemEdit hinzüfügentMenuItem # tMenuItemEdit->WinMenuItemA
```



## Kontakt

obj ->



**WinMenuItemRemove([logic1])**

**Menüeintrag aus Menü entfernen**

**obj     Menüeintrag**

**Menüeintrag**

**logic1 leeren**

**(optional)**

**Verwande**

**Siehe Befehle**

Dieser Befehl entfernt ein Menüeintrag (obj) inklusive aller Untermenüeinträge aus einem Menü. Wird der optionalen Parameter (logic1) auf true gesetzt, werden nur die Untermenüeinträge des Menüeintrags (obj) entfernt.



Bei einem Kontextmenü muss das Entfernen eines Menüeintrags in dem Ereignis EvtMenuInitPopup vorgenommen werden.

**Beispiel:**

```
// Wenn Menüeintrag vorhanden if (tMenuItem > 0){ // Menüeintrag entfernen tMenuItem->WinMenuIte
```



## Kontakt

In beiden Modi werden die Ereignisse EvtMenuContext und EvtMenuInitPopup des übergebenen Objektes durchgeführt, sofern sie dort eingetragen sind.

### Resultate:

Findet die Verarbeitung des ausgewählten Menüpunkts in der EvtMenuCommand statt, wird immer ErrOk zurückgegeben.

Wird der Modus WinMenuContextReturnID verwendet, wird nach dem Anklicken des Menüeintrags dessen MenuId zurückgegeben. Ist die MenuId negativ, wird der Wert in eine positive Zahl gewandelt, um Konflikte mit Fehlerwerten zu vermeiden.

Bei der Rückgabe von ErrUnavailable konnte das in (alpha1) angegebene Menü nicht gefunden werden. Verfügt der Benutzer nicht über ausreichende Rechte, um das Menü auszuführen, wird ErrRights zurückgegeben.

### Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	In (obj) wurde kein Oberflächen-Objekt übergeben.
<u>ErrValueInvalid</u>	Bei dem übergebenen Objekt ist die Eigenschaft <u>MenuNameCntxt</u> leer oder nicht vorhanden und in (alpha1) wurde kein Menü übergeben.

## RecList-Befehle

Befehle zum Bearbeiten eines RecList-Objekts

Verwandte

Befehle,

Siehe RecList,

Befehlsgruppen,

Befehlsliste

Befehle

- WinLstEdit

obj -> WinLstEdit(int1[, int2])  
: handle



Feld in einer Liste editieren

obj        Deskriptor der Liste  
int1       Deskriptor der Spalte

	Optionen	
	<u>WinLstEditLst</u>	Edit-Objekt
		mit Liste
	<u>WinLstEditLstAlpha</u>	Edit-Objekt
int2		mit
		zweispaltiger
		Liste
	<u>WinLstEditClearChanged</u>	Changed-Flag
		zurücksetzen

Resultat handle                      Objekt  
Verwandte Befehle, EvtLstEditStart,  
EvtLstEditCommit,

Siehe

EvtLstEditFinished, Ereignisabläufe  
WinLstEdit(), Beispiel

Mit diesem Befehl kann ein Feld innerhalb einer RecList oder DataList editiert werden.

In (obj) und (int1) muss der Deskriptor der Liste und der Spalte übergeben werden. Als Resultat wird der Deskriptor auf das Edit-Objekt zurückgegeben, der von diesem Befehl erzeugt wurde. Das Objekt ist vom gleichen Typ wie die Spalte.

Innerhalb des Listen-Objekts muss eine Zeile selektiert sein. Bei einem DataList-Objekt muss also die Eigenschaft CurrentInt auf eine Zeilennummer gesetzt sein.

In Abhängigkeit von den angegebenen Optionen in (int2) werden neben dem Eingabeobjekt noch weitere Objekte erzeugt:

- 0

Es werden keine weiteren Objekte erzeugt. Das Feld kann editiert werden.

- WinLstEditLst

Es wird ein DataListPopup-Objekt mit einer Spalte erzeugt. Die Spalte hat den gleichen Typ, wie das zu editierende Feld.

- WinLstEditLstAlpha

Es wird ein DataListPopup-Objekt mit zwei Spalten erzeugt. Die erste Spalte hat den gleichen Typ, wie das zu editierende Feld. Die zweite Spalte ist vom Typ alpha und kann mit Informationen zum Eintrag in der ersten Spalte genutzt werden. Der Wert der ersten Spalte wird in das Feld übertragen.

Die angegebenen Optionen können mit der Option WinLstEditClearChanged kombiniert werden. Diese Option setzt das Changed-Flag zurück. Das Flag wird nach dem Ereignis EvtLstEditCommit gesetzt, wenn die Funktion true zurückgegeben hat.

## Kontakt

Das Flag wird in dem Ereignis EvtLstEditFinished mit übergeben, um kenntlich zu machen, dass Änderungen an den Daten vorgenommen wurden.



Das Flag bleibt so lange gesetzt, bis es durch einen Aufruf von `WinLstEdit()` mit der Option WinLstEditClearChanged zurückgesetzt wird.

Bei Verwendung einer DataList ist folgendes zu beachten:

Standardmäßig wird nach Übernahme eines Eintrages aus der Liste im Edit-Objekt verblieben. Soll das Edit-Objekt verlassen und damit die Ereignisse EvtLstEditCommit und EvtLstEditFinished ausgelöst werden, muss in dem Ereignis EvtLstEditStart die Eigenschaft LstFlags des DataListPopup-Objektes auf WinLstEditClose gesetzt werden.

Beispiel:

```
sub EvtMouseEvent( aEvt      : event;      // Ereignis  aButton    : int;          // Maustaste  aHitTes
```

## RecView-Befehle

Befehle eines RecView-Objekts

Verwandte

Befehle,

Siehe RecView,

Befehlsgruppen,

Befehlsliste

Befehle

- WinRvwColumn
- WinRvwEdit
- WinRvwUpdate

## Kontakt

obj -> WinRvwColumn(point1[, int2]) : int;



Spalte eines RecView von einer Koordinate ermitteln

obj RecView-Objekt

point1 Position relativ zur linken oberen Ecke  
des RecView-Objektes

int2 Optionen (optional)  
\_WinRvwColumnHitHeader Spaltenköpfe

beachten

Resultat int Spaltendeskriptor oder 0



Siehe Verwandte Befehle

Mit diesem Befehl ist es möglich, ausgehend von einer gegebenen Koordinate (point1), den Deskriptor der Spalte zu ermitteln. Dies ist insbesondere im Ereignis EvtMouseEvent hilfreich, wenn dort der Spaltendeskriptor benötigt wird.

Der Rückgabewert liefert den Deskriptor der Spalte an der gegebenen Position oder 0. Ein Deskriptor wird zurückgegeben, wenn die Position (point1) innerhalb des Anzeigebereiches des RecView liegt (ohne Spaltenköpfe und ohne Scrollbars). Sollen die Spaltenköpfe berücksichtigt werden, dann kann die Option (int2) \_WinRvwColumnHitHeader angegeben werden.

Beispiel:

```
sub EvtMouseEvent( aEvt : event; // Ereignis aButton : int;
```

Mögliche Laufzeitfehler:

\_ErrHdlInvalid In (obj) wurde kein gültiger Deskriptor eines RecView-Objektes angegeben.

\_ErrValueInvalid In (int2) wurde eine ungültige Option angegeben.



## Kontakt

obj -> WinRvwEdit(int1[, int2[, int3[, int4]]) :



int

(Sub-)Item eines RecView-Objektes bearbeiten

obj      Objekt  
         Nummer des

int1      Items  
         Nummer des

int2      Subitems  
         Nummer des

int3      Views

int4      Optionen      —

Resultat int Fehlerwert

Siehe      Verwandte

Befehle, Blog

Dieser Befehl startet die Bearbeitung eines Items (int1) oder SubItems (int2) in dem View (int3) des RecView-Objektes (obj) durch. Wird kein SubItem (int2) angegeben, oder 0 übergeben, wird das übergeordnete Item bearbeitet.



Die Gruppe des zu bearbeitenden Items muss selektiert und sichtbar sein. Daher empfiehlt sich vorher die Verwendung von WinRvwUpdate() mit den Update-Modi WinRvwUpdateFromSelected und WinRvwUpdateDoKeepSelect.

Durch den Befehl wird der Ablauf, wie in Ereignisabläufe des RecViews beschrieben, durchgeführt. Wird daher im Ereignis EvtLstEditStartItem das Resultat WinRvwGroupEditSkipItem gesetzt, wird das nächste zu bearbeitende Item gesucht.

Die Nummer des Views (int3) gibt an, in welchem View die Gruppe bearbeitet werden soll. Mögliche Werte sind 0 bis 4. Wird 0 angegeben, oder das Argument weggelassen, wird im aktiven View, welches den Eingabe-Fokus besitzt, bearbeitet. Eine Angabe von 0 und 1 ist immer möglich. Die Views 2 bis 4 existieren nur, wenn eine entsprechende Splittung des RecView durch den Anwender vorgenommen wurde. Existiert das angegebene View nicht wird der Fehlerwert ErrUnknown zurückgegeben.

Als Option (int4) kann WinRvwEditAbortEditing angegeben werden, um eine aktuell aktive Bearbeitung abubrechen. Diese Option ist standardmäßig aktiv. Soll die Bearbeitung nicht abgebrochen werden, muss 0 übergeben werden. Ist in diesem Fall bereits ein Datensatz in Bearbeitung, wird der Fehlerwert ErrInUse zurückgegeben.

Als Rückgabewert wird der Fehlerwert ErrUnknown zurückgegeben, wenn das angegebene View (int3) nicht existiert oder noch nicht initialisiert (siehe EvtLstViewInit) wurde. Wurde als Option (int4) 0 übergeben und es ist bereits ein Datensatz in Bearbeitung wird ErrInUse zurückgegeben. Ist das Item (int1) oder das SubItem (int2) nicht vorhanden, kein Datensatz markiert, oder der aktive Datensatz nicht im sichtbaren Bereich, gibt der Befehl den Fehlerwert ErrUnavailable zurück. Ist keines der EvtLstEdit...-Ereignisse bei dem RecView (obj) eingetragen, kommt der Fehlerwert ErrIllegalOp. Sonst wird ErrOk zurückgegeben.

### Mögliche Laufzeitfehler

ErrHdlInvalid

In (obj) wurde kein gültiger Deskriptor angegeben.

## Kontakt

**ErrValueInvalid** In den Optionen (int4) wurde weder 0 noch  
**WinRvwEditAbortEditing** angegeben.

## Kontakt

obj -> WinRvwUpdate([int1[, int2[, int3]]) :



int

View eines RecView-Objektes aktualisieren

obj      Objekt  
          Update-Modus

int1  
          (siehe Text)  
          Nummer des

int2  
          Views  
          Datensatzpuffer

int3  
          oder  
          Dateinummer

Resultat int Fehlerwert 

Verwandte

Siehe    Befehle,  
          WinUpdate()

Dieser Befehl führt eine Positionierung in dem View (int2) des RecView-Objektes (obj) durch.

Folgende Parameter können als Update-Modus (int1) übergeben werden:

<u>_WinRvwUpdateFromFirst</u>	Neuaufbau des RecViews ab dem ersten Datensatz
<u>_WinRvwUpdateFromLast</u>	Neuaufbau des RecViews ab dem letzten Datensatz
<u>_WinRvwUpdateFromRecBuf</u>	Neuaufbau des RecViews ausgehend von den Werten des Feldpuffers (int3)
<u>_WinRvwUpdateFromTop</u>	Neuaufbau des RecViews ab dem ersten sichtbaren Datensatz
<u>_WinRvwUpdateFromTopLocation</u>	Neuaufbau der angezeigten Datensätze des RecViews. Die Position der ersten angezeigten Gruppe wird beibehalten.
<u>_WinRvwUpdateFromSelected</u>	Neuaufbau des RecViews ab dem selektieren Datensatz
<u>_WinRvwUpdateDoSelect</u>	In dem RecView wird der aktuelle Datensatz selektiert
<u>_WinRvwUpdateDoKeepSelect</u>	In dem RecView wird der aktuell selektierte Datensatz weiterhin selektiert
<u>_WinRvwUpdateOptClearCache</u>	Alle Gruppen werden aus dem Cache des RecView-Objektes entfernt
<u>_WinRvwUpdateOptClearSelected</u>	Die selektierte Gruppe wird aus dem Cache des RecView- Objektes entfernt

Konstanten aus den Bereichen \_WinRvwUpdateFrom\*, \_WinRvwUpdateDo\* und \_WinRvwUpdateOpt\* können miteinander kombiniert werden. Konstanten aus gleichen Bereichen können nicht kombiniert werden.



Wird zwar ein Update-Modus (int1), aber keine Konstante aus dem Bereich \_WinRvwUpdateFrom\* angegeben, wird der Laufzeitfehler ErrValueInvalid ausgelöst.

Wird als Update-Modus (int1) \_WinRvwUpdateFromRecBuf übergeben, muss in int3 ein Datensatzpuffer oder eine Datei übergeben werden. Diese muss mit der

## Kontakt

Eigenschaft DbFileNo oder DbLinkFileNo übereinstimmen. Wird in (int1) nicht WinRvwUpdateFromRecBuf angegeben, so führt eine Angabe von einem Datensatzpuffer oder einer Datei zu dem Laufzeitfehler ErrValueInvalid. Dieser Modus kann auch verwendet werden, um verknüpfte Datensätze anzuzeigen. Dazu wird der Quelldatensatz gelesen und die Quelldatei in int3 übergeben. Um in einer Verknüpfung auf einen bestimmten Datensatz zu positionieren, muss bei dem Verknüpfungsfeld "nur Zugriffspositionierung" aktiviert sein.

Fehlt die Angabe des Update-Modus, so wird WinRvwUpdateFromTop verwendet.

In der Nummer des Views (int2) wird definiert, welche Anzeige aktualisiert werden soll. Mögliche Werte sind 0 bis 4. Wird 0 angegeben, oder das Argument weggelassen, wird das aktive View, welches den Eingabe-Fokus besitzt, aktualisiert. Eine Angabe von 0 und 1 ist immer möglich. Die Views 2 bis 4 existieren nur, wenn eine entsprechende Splittung des RecView durch den Anwender vorgenommen wurde. Existiert das angegebene View nicht wird der Fehlerwert ErrUnknown zurückgegeben.

Der Befehl hat zur Folge, dass die Anzeige des entsprechenden Views neu aufgebaut wird. Daher werden anschließend EvtLstGroupInit-Ereignisse durchgeführt.

Als Rückgabewert wird der Fehlerwert ErrUnknown zurückgegeben, wenn das angegebene View (int2) nicht existiert oder noch nicht initialisiert (siehe EvtLstViewInit) wurde. Der Fehlerwert ErrUnavailable wird zurückgegeben, wenn der Update-Modus WinRvwUpdateFromTop angegeben wurde und kein Datensatz angezeigt wird, oder der Update-Modus WinRvwUpdateFromSelected verwendet wird und kein Datensatz selektiert ist. Sonst wird ErrOk zurückgegeben.



Wird der Befehl in einem der Ereignisse EvtLstViewInit, EvtLstGroupInit, EvtLstEditStartGroup, EvtLstEditStartItem, EvtLstEditActivate oder EvtLstEditEndItem aufgerufen, wird der Laufzeitfehler ErrHdlInvalid ausgelöst.  
Mögliche Laufzeitfehler:

ErrHdlInvalid

In (obj) wurde kein gültiger Deskriptor angegeben. Wird beim Update-Modus (int1) WinRvwUpdateFromRecBuf kein Datensatzpuffer oder die Nummer oder ein Datensatzpuffer einer anderen Datei angegeben, wird der Laufzeitfehler ebenfalls ausgelöst. Weiterhin wird der Laufzeitfehler ausgelöst, wenn der Befehl in einem der oben genannten Ereignisse aufgerufen wird.

In (int3) wurde ein Wert angegeben und der Update-Modus in (int1) ist nicht WinRvwUpdateFromRecBuf oder es wurde ein ungültiger ErrValueInvalid Update-Modus angegeben. Wird in (int1) keine Konstante aus dem Bereich \_WinRvwUpdateFrom\* angegeben, wird der Laufzeitfehler ebenfalls ausgelöst.

## RtfEdit-Befehle

### Befehle und Konstanten für RtfEdit-Objekt

#### Verwandte

#### Befehle,

Siehe RtfEdit,

#### Befehlsgruppen,

#### Befehlsliste

### Befehle

- RtfTabMake
- WinEmfProcess
- WinRtfLoad
- WinRtfLoadBin
- WinRtfLoadName
- WinRtfPicInsertMem
- WinRtfPicInsertName
- WinRtfSave
- WinRtfSaveBin
- WinRtfSaveName
- WinRtfSearch
- WinRtfTabGet
- WinRtfTabSet

### Konstanten

- WinRtfLoadAscii
- WinRtfLoadAuto
- WinRtfLoadInsert
- WinRtfLoadMix
- WinRtfLoadOem
- WinRtfLoadRtf
- WinRtfPicModeAuto
- WinRtfPicModeQuality
- WinRtfPicModeSpeed
- WinRtfSaveAscii
- WinRtfSaveAuto
- WinRtfSaveMark
- WinRtfSaveMix
- WinRtfSaveOem
- WinRtfSaveRtf
- WinRtfSearchCase
- WinRtfSearchDelete
- WinRtfSearchReplace
- WinRtfSearchUp
- WinRtfSearchWord
- WinRtfTabCenter
- WinRtfTabDecimal
- WinRtfTabLeft
- WinRtfTabNone
- WinRtfTabRight
- WinStreamBufField

- WinStreamBufText
- WinStreamCaption
- WinStreamNameBin
- WinStreamNameFile
- WinStreamNameText

obj -> WinRtfLoad(int1[,  
int2[, handle3]]) : int



Text in RtfEdit-Objekt laden

obj      Objekt (RtfEdit-Objekt)

Quelle des Textes

WinStreamCaption Text aus der  
Eigenschaft  
Caption

int1      WinStreamBufField Text aus

Feldpuffer  
laden

WinStreamBufText Text aus  
Textpuffer  
laden

Modus (optional)

WinRtfLoadAuto      Textformat  
automatisch

WinRtfLoadRtf      erkennen  
RTF-Text

int2      WinRtfLoadAscii      laden  
ASCII-Text

WinRtfLoadInsert Text in den  
bestehenden  
Text einfügen

WinRtfLoadMix      Text mit  
Daten  
mischen

int3      Deskriptor des Textpuffers  
(optional)

Resultat int      Fehlerwert

Verwandte Befehle,

Siehe WinRtfLoadName(),  
WinRtfLoadBin(), WinRtfSave()

Mit diesem Befehl wird ein Text in ein RtfEdit-Objekt geladen. Der Deskriptor des Objektes wird in (obj) übergeben.

In (int1) wird die Quelle des zu ladenden Textes definiert. Der Deskriptor der Quelle kann in (handle3) angegeben werden. Wird kein Deskriptor angegeben, wird der Inhalt der entsprechenden Eigenschaft verwendet.

Folgende Quellen können angegeben werden:

- WinStreamCaption

Werden prozedural Änderungen der Caption vorgenommen, kann mit dieser Option eine Aktualisierung des Objektes durchgeführt werden. Der Parameter (handle3) darf nicht gesetzt werden.

## Kontakt

- WinStreamBufField

Der Inhalt des in der Eigenschaft DbFieldName angegebenen Feldpuffers wird geladen. der Parameter (handle3) darf nicht gesetzt sein.

- WinStreamBufText

Der Text steht in einem Textpuffer bereit. Der Deskriptor des Textes kann in (handle3) angegeben werden. Ist kein Textpuffer angegeben, wird der in der Eigenschaft DbTextBuf angegebene Textpuffer verwendet.

Der Parameter (int2) bestimmt das Format der Quelle. Folgende Konstanten können übergeben werden:

- WinRtfLoadAuto

Das Format der Quelle wird automatisch bestimmt.

- WinRtfLoadRtf

Die Textquelle ist im RTF-Format.

- WinRtfLoadAscii

Die Textquelle ist im ASCII-Format.

Die Parameter zum Quellenformat können mit WinRtfLoadInsert kombiniert werden, um in einen bestehenden Text den angegebenen Text einzufügen.

Der Text ersetzt den Text, der mit der Markierung (Range) selektiert ist. Ist keine Markierung vorhanden, wird der Text an der aktuellen Cursorposition eingefügt.

Bei der Kombination mit der Option WinRtfLoadMix werden beim Laden des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.

Als Rückgabewert können neben den Fehlerkonstanten aus dem Bereich der externe Dateien die Werte ErrData und ErrRtfSyntaxError zurückgegeben werden. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.





obj -> WinRtfLoadBin(handle1, int2[,  
alpha3]) : int

Binäres Objekt in RtfEdit-Objekt laden  
obj            Objekt (RtfEdit-Objekt)

handle1 Deskriptor des binären Objektes

	Modus (optional)	
	<u>WinRtfLoadAuto</u>	Textformat automatisch erkennen
	<u>WinRtfLoadRtf</u>	RTF-Text laden
int2	<u>WinRtfLoadAscii</u>	ASCII-Text laden
	<u>WinRtfLoadInsert</u>	Text in den bestehenden Text einfügen
	<u>WinRtfLoadMix</u>	Text mit Daten mischen

alpha3    Verschlüsselungscode (optional)

Resultat int    Fehlerwert

Verwandte Befehle,

Siehe WinRtfLoad(), WinRtfSaveBin(),  
BinOpen()

Mit diesem Befehl wird der Inhalt eines binären Objekts in ein RtfEdit-Objekt geladen. Der Deskriptor des RtfEdit-Objektes wird in (obj), der Deskriptor des binären Objekts in (handle1) übergeben.

Der Parameter (int2) bestimmt das zu lesende Format. Folgende Konstanten können angegeben werden:

- WinRtfLoadAuto

Das Format der Quelle wird automatisch bestimmt.

- WinRtfLoadRtf

Die Textquelle ist im RTF-Format.

- WinRtfLoadAscii

Die Textquelle ist im ASCII-Format.

Die Parameter zum Quellenformat können mit WinRtfLoadInsert kombiniert werden, um in einen bestehenden Text den angegebenen Text einzufügen.

Der Text ersetzt den Text, der mit der Markierung (Range) selektiert ist. Ist keine Markierung vorhanden, wird der Text an der aktuellen Cursorposition eingefügt.

Bei der Kombination mit der Option WinRtfLoadMix werden beim Laden des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen

## Kontakt

befinden sich im Abschnitt Text und Daten mischen.

Als Rückgabewert können neben den Fehlerkonstanten aus dem Bereich der externe Dateien die Werte \_ErrData und \_ErrRtfSyntaxError zurückgegeben werden. Bei der Rückgabe von \_ErrOk ist kein Fehler aufgetreten.

obj ->

**WinRtfLoadName(int1,**



**int2[, alpha3])) : int**

**Text in RtfEdit-Objekt laden**

**obj**      **Objekt (RtfEdit-Objekt)**

**Quelle des Textes**

**\_WinStreamNameText** internen

**Text**

**int1**

**laden**

**\_WinStreamNameFile** externen

**Text**

**laden**

**Modus**

**\_WinRtfLoadAuto**      **Textformat  
automatisch  
erkennen**

**\_WinRtfLoadRtf**      **RTF-Text  
laden**

**\_WinRtfLoadAscii**      **ASCII-Text  
laden**

**int2**

**\_WinRtfLoadOem**      **OEM-Text  
laden**

**\_WinRtfLoadInsert** **Text in den  
bestehenden  
Text einfügen**

**\_WinRtfLoadMix** **Text mit  
Daten  
mischen**

**alpha3**      **Name des Textes oder Feldpuffer  
(optional)**

**Resultat int**      **Fehlerwert**

**Verwandte Befehle,**

**Siehe**      **WinRtfLoad(), WinRtfLoadBin(),  
WinRtfSaveName()**

Mit diesem Befehl wird ein Text in ein **RtfEdit**-Objekt geladen. Der Deskriptor des Objektes wird in (obj) übergeben.

In (int1) wird die Quelle des zu ladenden Textes definiert. Der Name der Quelle kann in (alpha3) angegeben werden. Wird kein Name angegeben, wird der Inhalt der entsprechenden Eigenschaft verwendet.

Folgende Quellen können angegeben werden:

- **\_WinStreamNameText**

Der Text steht in einem internen Text zur Verfügung. Der Name des Textes kann entweder in (alpha3) übergeben werden oder wird der Eigenschaft **FileName** entnommen.

## Kontakt

- WinStreamNameFile

Der Text steht in einer externen Datei zur Verfügung. Der Name der Datei kann entweder in (alpha3) übergeben werden oder wird der Eigenschaft FileName entnommen.

Der Parameter (int2) bestimmt das Format der Quelle. Folgende Konstanten können übergeben werden:

- WinRtfLoadAuto

Das Format der Quelle wird automatisch bestimmt.

- WinRtfLoadRtf

Die Textquelle ist im RTF-Format.

- WinRtfLoadAscii

Die Textquelle ist im ASCII-Format.

- WinRtfLoadOem

Die Textquelle ist im OEM-Format.

Die Parameter zum Quellenformat können mit WinRtfLoadInsert kombiniert werden, um in einen bestehenden Text den angegebenen Text einzufügen.

Der Text ersetzt den Text, der mit der Markierung (Range) selektiert ist. Ist keine Markierung vorhanden, wird der Text an der aktuellen Cursorposition eingefügt.

Bei der Kombination mit der Option WinRtfLoadMix werden beim Laden des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.

Als Rückgabewert können neben den Fehlerkonstanten aus dem Bereich der externe Dateien die Werte ErrData und ErrRtfSyntaxError zurückgegeben werden. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.

## Kontakt

obj -> WinRtfSave(int1[, int2[,  
int3]]) : int



Text aus RtfEdit-Objekt speichern

obj Objekt (RtfEdit-Objekt)

Ziel des Textes

WinStreamCaption Text in die

Eigenschaft

Caption

schreiben

WinStreamBufField Text in

int1

einen

Feldpuffer

schreiben

WinStreamBufText Text in

einen

Textpuffer

schreiben

Modus (optional)

WinRtfSaveAuto Textformat

automatisch

erkennen

WinRtfSaveRtf

RTF-Text

sichern

int2

WinRtfSaveAscii ASCII-Text

sichern

WinRtfSaveMark markierten

Text sichern

WinRtfSaveMix Text mit Daten

mischen

Deskriptor des Textpuffers

int3

(optional)

Resultat int Fehlerwert

Verwandte Befehle,

Siehe WinRtfSaveName(),

WinRtfSaveBin(), WinRtfLoad()

Mit diesem Befehl wird ein Text aus einem RtfEdit-Objekt gespeichert. Der Deskriptor des Objektes wird in (obj) übergeben.

In (int1) wird das Ziel des Textes definiert. Der Deskriptor des Ziels wird in (int3) angegeben werden.

Folgende Ziele können angegeben werden:

- WinStreamCaption

Werden prozedural Änderungen im Text vorgenommen, kann mit dieser Option eine Aktualisierung des Objekts durchgeführt werden. Der Parameter (int3) darf nicht gesetzt werden.

## Kontakt

- WinStreamBufField

Der Text wird in den in der Eigenschaft DbFieldName angegebenen Feldpuffer gespeichert. Der Parameter (int3) darf nicht gesetzt sein.

- WinStreamBufText

Der Text wird in den in (int3) übergebenen Textpuffer geschrieben. Ist kein Textpuffer angegeben, wird der in der Eigenschaft DbTextBuf angegebene Textpuffer verwendet.

Der Parameter (int2) bestimmt das zu schreibende Format. Folgende Konstanten können übergeben werden:

- WinRtfSaveAuto

Das Format des Textes wird automatisch bestimmt.

- WinRtfSaveRtf

Der Text wird im RTF-Format geschrieben.

- WinRtfSaveAscii

Der Text wird im ASCII-Format geschrieben.

Die Parameter zum Zielformat können mit WinRtfSaveMark kombiniert werden, um einen markierten Textbereich innerhalb des Textes zu sichern.

Bei der Kombination mit der Option WinRtfSaveMix werden beim Speichern des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.

Als Rückgabewert können neben den Fehlerkonstanten aus dem Bereich der externe Dateien die Werte ErrData und ErrRtfSyntaxError zurückgegeben werden. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.

## Kontakt

obj -> WinRtfSaveBin(int1, int2[, int3[, alpha4]]) : int  Text aus

RtfEdit-Objekt in binärem Objekt speichern

obj      Objekt (RtfEdit-Objekt)

int1      Deskriptor des binären  
          Objektes

Modus (optional)

WinRtfSaveAuto      Textformat  
                          automatisch

WinRtfSaveRtf      erkennen  
                          RTF-Text

int2      WinRtfSaveAscii      sichern  
                                  ASCII-Text

WinRtfSaveMark      sichern  
                          markierten  
WinRtfSaveMix      Text sichern  
                          Text mit  
                          Daten  
                          mischen

int3      Kompressionsfaktor (optional)  
          Verschlüsselungscode

alpha4    (optional)

Resultat int      Fehlerwert

Verwandte Befehle,

Siehe WinRtfSave(), WinRtfLoadBin(),  
BinOpen()

Mit diesem Befehl wird ein Text aus einem RtfEdit-Objekt in einem binären Objekt gespeichert. Der Deskriptor des RtfEdit-Objektes wird in (obj), der Deskriptor des binären Objekts in (int1) übergeben.

Der Parameter (int2) bestimmt das zu schreibende Format. Folgende Konstanten können angegeben werden:

- WinRtfSaveAuto

Das Format des Textes wird automatisch bestimmt.

- WinRtfSaveRtf

Der Text wird im RTF-Format geschrieben.

- WinRtfSaveAscii

Der Text wird im ASCII-Format geschrieben.

Die Parameter zum Zielformat können mit WinRtfSaveMark kombiniert werden, um einen markierten Teil innerhalb des Textes zu sichern.

Bei der Kombination mit der Option WinRtfSaveMix werden beim Speichern des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.

## Kontakt

Als Rückgabewert können neben den Fehlerkonstanten aus dem Bereich der externe Dateien die Werte \_ErrData und \_ErrRtfSyntaxError zurückgegeben werden. Bei der Rückgabe von \_ErrOk ist kein Fehler aufgetreten.



## Kontakt

obj -> WinRtfSaveName(int1, int2,  
alpha3) : int



Text aus RtfEdit-Objekt speichern

obj Objekt (RtfEdit-Objekt)

Ziel des Textes

WinStreamNameText Text in

einen

internen

Text

schreiben

int1

WinStreamNameFile Text in

eine

externe

Datei

schreiben

Modus

WinRtfSaveAuto

Textformat  
automatisch

WinRtfSaveRtf

erkennen  
RTF-Text

WinRtfSaveAscii

sichern  
ASCII-Text

int2

WinRtfSaveOem

sichern  
OEM-Text

WinRtfSaveMark

sichern  
markierten

WinRtfSaveMix

Text sichern  
Text mit Daten  
mischen

alpha3 Zielname

Resultat int Fehlerwert

Verwandte Befehle, WinRtfSave(),

Siehe WinRtfSaveBin(),

WinRtfLoadName()

Mit diesem Befehl wird ein Text aus einem RtfEdit-Objekt gespeichert. Der Deskriptor des Objektes wird in (obj) übergeben.

In (int1) wird das Ziel des Textes definiert. Der Name des Ziels wird in (alpha3) angegeben.

Folgende Ziele können angegeben werden:

- WinStreamNameText

Der Text wird in einem internen Text gespeichert. Der Name des Textes wird in (alpha3) übergeben.

- WinStreamNameFile

## Kontakt

Der Text wird in einer externen Datei gespeichert. Der Name der Datei wird in (alpha3) übergeben.

Der Parameter (int2) bestimmt das zu schreibende Format. Folgende Konstanten können übergeben werden:

- WinRtfSaveAuto

Der Text wird im RTF-Format geschrieben.

- WinRtfSaveRtf

Der Text wird im RTF-Format geschrieben.

- WinRtfSaveAscii

Der Text wird im ASCII-Format geschrieben.

- WinRtfSaveOem

Der Text wird im OEM-Format geschrieben.

Die Parameter zum Zielformat können mit WinRtfSaveMark kombiniert werden, um einen markierten Teil innerhalb des Textes zu sichern.

Bei der Kombination mit der Option WinRtfSaveMix werden beim Speichern des Textes die Platzhalter durch die entsprechenden Daten ersetzt. Weitere Informationen befinden sich im Abschnitt Text und Daten mischen.

Als Rückgabewert können neben den Fehlerkonstanten aus dem Bereich der externe Dateien die Werte ErrData und ErrRtfSyntaxError zurückgegeben werden. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.

## Kontakt

obj -> WinRtfSearch(alpha1[, int2[, range3[, alpha4[, var  
alpha5]]]]) : int



Zeichenfolge in einem Text suchen / Suchen und Ersetzen  
obj Deskriptor eines RtfEdit-Objekts

alpha1	Suchtext	
	Optionen (optional)	
	<u>WinRtfSearchUp</u>	Suchbereich vom Ende nach Vorne
	<u>WinRtfSearchCase</u>	durchsuchen Groß-/Kleinschreibung
int2	<u>WinRtfSearchWord</u>	beachten Nur ganze Wörter
	<u>WinRtfSearchReplace</u>	suchen Suchbegriff durch
	<u>WinRtfSearchDelete</u>	(alpha4) ersetzen Bereich in (range3)
		entfernen
range3	Suchbereich (optional)	
alpha4	Ersetzungstext (optional)	
var	Auf Suchtext folgende Zeichenkette	
<u>alpha5</u>	(optional)	
Resultat	int Position der gefundenen Zeichenfolge	
Siehe	<u>Verwandte Befehle, PrtRtfSearch(),</u> <u>TextSearch()</u>	

Diese Funktion durchsucht einen Text in einem RtfEdit-Objekt. Der Deskriptor des Objektes wird in (obj) angegeben.

Die zu suchende Zeichenkette wird in (alpha1) angegeben. Alle weiteren Parameter sind optional. Werden keine weiteren Parameter angegeben, wird der gesamte Text von vorne nach hinten nach der Zeichenkette durchsucht. Wird der Begriff gefunden, wird die Position des ersten Zeichens innerhalb des Textes zurückgegeben. Befindet sich die zu suchende Zeichenkette mehrfach im Text, wird nur das erste Vorkommen zurückgegeben. Ist die Zeichenkette nicht vorhanden wird der Wert -1 zurückgegeben.

### Beispiel:

```
tPos # $RtfEdit->WinRtfSearch('suche');
```

Die Suche kann mit folgenden Optionen beeinflusst werden:

- WinRtfSearchUp

Der Suchbereich wird vom Ende zum Anfang durchsucht.

- WinRtfSearchCase

Die Groß- und Kleinschreibung des Suchbegriffes wird beachtet.

- WinRtfSearchWord

## Kontakt

Es wird nur nach ganzen Wörtern gesucht.

- WinRtfSearchReplace

Der Suchtext wird durch den Ersetzungstext in (alpha4) ersetzt.

- WinRtfSearchDelete

Der Bereich in (range3) wird aus dem RTF-Text entfernt.

Die Optionen können miteinander kombiniert werden. Der zu durchsuchende Text kann durch die Angabe eines Bereiches in (range3) bestimmt werden. Standardmäßig wird der Bereich (0, -1) (Anfang bis Ende) durchsucht.

Wird die Option WinRtfSearchReplace angegeben, muss in (alpha4) ein entsprechender Ersetzungstext angegeben werden, da sonst der Suchbegriff aus dem Text entfernt wird. Wie beim Suchen wird nur die erste Fundstelle ersetzt.

Wird die Option WinRtfSearchDelete angegeben, kann zusätzlich mit der Option WinRtfSearchReplace der Bereich durch einen anderen Text ersetzt werden. Der entsprechende Ersetzungstext wird in (alpha4) angegeben.

Wird der optionale var-Parameter (alpha5) angegeben, dann wird in dieser Variable der Text hinterlegt, der hinter dem Suchtext (alpha1) steht. Wird der Suchtext nicht gefunden, ist die Variable nach dem Aufruf leer. Die Länge des Nachfolgenden Textes richtet sich nach der Dimension der Variable. Bei einem alpha(10) beispielsweise, werden maximal 10 Zeichen zurückgegeben.

### Beispiele:

```
// Suchen nach ganzem Wort mit Groß-/KleinschreibungPos # $RtfEdit->WinRtfSearch('Suche', _WinRt
```

### Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Optionen (int2) ungültig

## Kontakt

**obj -> WinRtfTabGet(int1[, var  
rtftab2]) : int**



**vorhandene Tabulatoren ermitteln**

**obj**            Deskriptor eines RtfEdit-Objekts

              Bereich

**int1**            WinEditAll        gesamtes Dokument

WinEditMark markierter Bereich

**var rtftab2** Array mit Tabulatoren

**int**    Anzahl der definierten

**Resultat**        Tabulatoren

**Siehe**            Verwandte Befehle, WinRtfTabSet()

Mit diesem Befehl werden alle Tabulatoren in einem RtfEdit-Objekt ermittelt.

In (int1) wird übergeben, ob die Tabulatoren für den gesamten Text ( WinEditAll) oder für den markierten Bereich ( WinEditMark) ermittelt werden.

Die Tabulatoren werden in einem Array vom Typ rtftab zurückgegeben.

Die Tabulatoren befinden sich in der Reihenfolge ihrer Position in dem Array. Die Anzahl der definierten Tabulatoren wird als Rückgabewert zurückgegeben.

**Beispiele:**

```
local{ tTab : rtftab[64];}...// Tabulatoren auslesenif ($rtfEdit->WinRtfTabGet(_WinEditAll, var
```

obj ->

**WinRtfTabSet**(int1[,        ])

**var** rtfstab2]) : int

**Tabulatoren setzen**

obj

Deskriptor eines

RtfEdit-Objekts

Bereich

WinEditAll

gesamtes

int1

Dokument

WinEditMark markierter

Bereich

var

Array mit Tabulatoren

rtfstab2

(optional)

Resultat

int

ErrOk

Verwandte Befehle,

Siehe

WinRtfTabGet()

Mit diesem Befehl können bis zu 64 Tabulatoren in einem RtfEdit-Objekt gesetzt werden.

In (int1) wird übergeben, ob die Tabulatoren für den gesamten Text ( WinEditAll) oder für den markierten Bereich ( WinEditMark) gesetzt werden.

Die Tabulatoren werden in einem Array vom Typ rtfstab übergeben und können mit dem Befehl RtfTabMake() definiert werden. Werden keine Tabulatoren übergeben, werden alle bestehenden Tabulatoren gelöscht.

Die Tabulatoren müssen in der Reihenfolge ihrer Position in dem Array angegeben sein.

Beispiele:

```
local{ tTab : rtfstab[64];}...// Tabulatoren definieren tTab[1]:tabpos # PrtUnitLog(2.0, _PrtUnit
```

Mit dem Befehl WinRtfTabSet() werden immer alle Tabulatoren übergeben. Um zu den vorhandenen Tabulatoren einen weiteren Tabulator zu setzen, müssen zunächst mit dem Befehl WinRtfTabGet() die vorhandenen Tabulatoren ermittelt und anschließend um den neuen Tabulator erweitert werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Deskriptor ist ungültig.

ErrFldType Das übergebene Array ist nicht vom Typ rtfstab.

## Kontakt

obj -> WinRtfPicInsertMem(handle1[, int2[,  
int3]]) : int



Einfügen von Bildern in ein RtfEdit-Objekt

obj        Deskriptor des RtfEdit-Objekts

handle1 Deskriptor des Memory-Objekts

int2        Seitennummer (optional)

Optionen (optional)

WinRtfPicModeSpeed        Performantes

Einfügen

WinRtfPicModeQuality        Qualitatives

Einfügen

int3        WinRtfPicModeAuto

Modus

abhängig von  
der Farbtiefe  
und Größe  
des Bildes  
auswählen

Resultat int Einfügeresultat (siehe Text)



Verwandte Befehle,

Siehe

WinRtfPicInsertName()

Diese Funktion fügt ein Bild aus einem Memory-Objekt (handle1) an der aktuellen Cursorposition eines RtfEdit-Objekts (obj) ein. Ist eine Selektion vorhanden, wird diese durch das Bild ersetzt. Das Argument Seitennummer (int2) bestimmt bei einem Multipage-TIFF, die Seite, die das einzufügende Bild enthält. Die Seitenzählung beginnt mit 1. Wird (int2) nicht angegeben oder ist der Wert Null, dann wird immer die erste Seite gewählt. Bei Formaten außer TIFF wird das Argument ignoriert.

Optional können in (int3) folgende Optionen angegeben werden:

WinRtfPicModeSpeed        Performantes Einfügen

WinRtfPicModeQuality        Qualitatives Einfügen

WinRtfPicModeAuto        Modus abhängig von der Farbtiefe und Größe des Bildes  
auswählen

Wird keine der Optionen angegeben, wird automatisch WinRtfPicModeAuto verwendet.

Die anzeigbaren Formate sind GIF, TIFF, JPEG, PNG und BMP.



Transparente oder semitransparente Pixel bei PNG bzw. 32bpp BMP-Dateien werden zu weißer Farbe gemischt.

Wird das Bildformat nicht unterstützt, gibt die Funktion den Fehlerwert ErrGeneric zurück. Ist der Arbeitsspeicher nicht ausreichend, wird ErrOutOfMemory zurückgegeben.

Beispiel:

```
// Bild ladentMem # MemAllocate(_MemAutoSize);tFsi # FsiOpen(_Sys->spPathMyPictures + '\MyPicture
```

Mögliche Laufzeitfehler:

## Kontakt

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein RtfEdit-Objekt



## Kontakt

obj -> WinRtfPicInsertName(alpha1[,  
alpha2[, int3[, int4]]]) : int



Einfügen von Bildern in ein RtfEdit-Objekt

obj      Deskriptor des RtfEdit-Objekts

alpha1    Name der Bilddatei

alpha2    Verschlüsselungs-Code (optional)

int3      Seitennummer (optional)

Optionen (optional)

WinRtfPicModeSpeed      Performantes

Einfügen

WinRtfPicModeQuality      Qualitatives

Einfügen

int4      WinRtfPicModeAuto

Modus

abhängig von  
der Farbtiefe

und Größe  
des Bildes  
auswählen

Resultat int Einfügeresultat (siehe Text)



Verwandte Befehle,

Siehe

WinRtfPicInsertMem()

Diese Funktion fügt ein Bild an der aktuellen Cursorposition eines RtfEdit-Objekts ein.

Ist eine Selektion vorhanden, wird diese durch das Bild ersetzt.

Durch die Voranstellung eines Präfixes (alpha1) bezeichnet der Name entweder eine externe Datei, ein binäres Objekt oder ein in der Datenbank gespeichertes Bild.

Der Verschlüsselungs-Code (alpha2) wird verwendet, wenn das Bild in der Datenbank verschlüsselt vorliegt. Das Argument Seitennummer (int3) bestimmt bei einem Multipage-Tiff, die Seite, die das einzufügende Bild enthält. Die Seitenzählung beginnt mit 1. Wird (int3) nicht angegeben oder ist der Wert Null, dann wird immer die erste Seite gewählt. Bei Formaten außer TIFF wird das Argument ignoriert.

Optional können in (int4) folgende Optionen angegeben werden:

WinRtfPicModeSpeed      Performantes Einfügen

WinRtfPicModeQuality      Qualitatives Einfügen

WinRtfPicModeAuto      Modus abhängig von der Farbtiefe und Größe des Bildes  
auswählen

Wird keine der Optionen angegeben, wird automatisch WinRtfPicModeAuto verwendet.

Die anzeigbaren Formate sind GIF, TIFF, JPEG, PNG und BMP.



Transparente oder semitransparente Pixel bei PNG bzw. 32bpp BMP-Dateien werden zu weißer Farbe gemischt.

Als Fehlerwerte können Fehlerwerte aus dem Bereich der Befehle für externe Dateien auftreten, wenn das Bild extern gelesen wird.

## Kontakt

Zusätzlich sind folgende Fehlerwerte möglich:

<u><b>ErrGeneric</b></u>	BLOB oder externe Datei enthält kein Bild oder ein unbekanntes Bildformat
<u><b>ErrNameInvalid</b></u>	Der Parameter (alpha1) ist leer
<u><b>ErrUnavailable</b></u>	Das Bild (alpha1) existiert nicht in der Datenbank
<u><b>ErrOutOfMemory</b></u>	Arbeitsspeicher nicht ausreichend

Beispiele:

```
// Externe Datei laden$RtfEdit->WinRtfPicInsertName('*' + _Sys->spPathMyPictures + '\MyPicture.jp
```

Mögliche Laufzeitfehler:

**ErrHdlInvalid** Bei (obj) handelt es sich nicht um ein RtfEdit-Objekt

**RtfTabMake(int1, int2) :**

**rtftab**



**Erzeugung eines Tabulators**

**int1      Tabulatorposition**

**Tabulatortyp**

**\_WinRtfTabNone**

**Tabulator löschen**

**\_WinRtfTabLeft**

**linksbündigen**

**\_WinRtfTabCenter**

**Tabulator setzen  
zentrierten**

**int2**

**\_WinRtfTabRight**

**Tabulator setzen  
rechtsbündigen**

**\_WinRtfTabDecimal** **Dezimal-Tabulator**

**Tabulator setzen  
setzen**

**Resultat rtftab Erzeugter Tabulator**

**Siehe Verwandte Befehle, rtftab**

**Dieser Befehl erzeugt aus den Angaben des Tabulatortyps und der Tabulatorposition eine rtftab-Struktur.**

**Die Tabulatorposition wird in logischen Einheiten angegeben.**

## Kontakt

obj -> WinEmfProcess(int1, alpha2[, int3[, int4[,  
int5]]]) : int



Text aus RtfEdit-Objekt als Bild speichern

obj RtfEdit-Objekt

Funktion

int1 \_WinEmfProcessCreate Bild erzeugen

\_WinEmfProcessDelete Bild löschen

alpha2 Name des Bildes oder der externen Datei

Optionen (optional)

\_WinEmfCreateOverwrite Zielfile oder -objekt

int3 überschreiben

\_WinEmfCreateDefault Zielfile oder -objekt

nicht überschreiben

int4 Position des Startzeichens (optional)

int5 Höhe des Bildes in Twips (optional)

int Position des ersten, nicht geschriebenen



Resultat Zeichens, 0 oder Fehlerwert

Siehe Verwandte Befehle

Der Befehl erstellt aus dem Inhalt eines RtfEdit-Objekts ein EMF-Bild. Das Bild kann als Storage-Objekt in der Datenbank oder als externe Datei gespeichert werden.

In (obj) wird der Deskriptor des RtfEdit-Objekts übergeben. Die durchzuführende Funktion ist in (int1) angegeben. Folgende Funktionen können durchgeführt werden:

- \_WinEmfProcessCreate (0)

Diese Funktion erzeugt das Bild. Der Name des erzeugten Objekts wird in (alpha2) angegeben. Wird dem Namen ein '\*' vorangestellt, erfolgt die Speicherung als externe Datei, sonst erfolgt die Speicherung im Storage-Verzeichnis MetaPicture. Wenn ein Storage-Objekt generiert wird, muss die Namensvergabe den Konventionen für Oberflächen-Objekte genügen (Eigenschaft Name), ansonsten den Konventionen für externe Dateinamen.

Über die Optionen (int3) kann bestimmt werden, ob ein vorhandenes Objekt bzw. eine vorhandene Datei überschrieben werden soll. Folgende Konstanten stehen zur Verfügung:

\_WinEmfCreateDefault (0) Bestehende externe Datei oder Bild-Objekt nicht überschreiben

\_WinEmfCreateOverwrite (1) Bestehende externe Datei oder Bild-Objekt überschreiben

Das Argument (int4) gibt an, ab welcher Zeichenposition des enthaltenen Textes die EMF-Ausgabe erfolgt. Fehlt das Argument oder ist es 0, wird am Textanfang begonnen.

(int5) definiert die Bildhöhe in logischen Einheiten, bis zu der der Text ausgegeben wird. Es werden alle Zeichen geschrieben, die vollständig dargestellt werden können. Der Befehl gibt anschließend das nächste Zeichen zurück, mit dem das nächste Bild startet. Wurde der gesamte Text formatiert,

## Kontakt

liefert der Befehl 0 zurück. Auf diese Weise kann der Text sukzessive in mehrere Bilder aufgeteilt werden. Wird die Bildhöhe nicht angegeben oder ist diese -1, wird der Text ab der Startposition bis zum Ende ausgegeben.

Die Breite des Bildes wird aus der Eigenschaft PageWidth des RtfEdit-Objekts entnommen.

- \_WinEmfProcessDelete (1)

Diese Funktion entfernt ein als Storage-Objekt erzeugtes Bild aus dem Verzeichnis MetaPicture. Das übergebene Objekt, sowie die Parameter (int3) bis (int5) werden ignoriert. Externe Dateien können mit der Anweisung FsiDelete() gelöscht werden.

Tritt beim Erzeugen des Bildes ein Fehler bei der Verarbeitung auf, wird ein negativer Wert zurückgegeben. Beim Löschen des Bildes wird im Fehlerfall ein Wert ungleich 0 zurückgegeben. Dieser Wert kann mit den folgenden Konstanten verglichen werden:

<u>_ErrNameInvalid</u>	Der übergebene Name ist leer oder ungültig.
<u>_ErrExists</u>	Die Datei oder das <u>Storage</u> -Objekt existiert bereits und die Option <u>_WinEmfCreateOverwrite</u> wurde nicht angegeben.
<u>_ErrFsi...</u>	Beim Schreiben in eine externe Datei können Fehler aus diesem Bereich auftreten.
<u>_ErrOutOfMemory</u>	Es steht nicht genügend Speicher zur Durchführung der Operation zur Verfügung.
<u>_rNoKey</u> Beispiel:	Das mit <u>_WinEmfProcessDelete</u> zu löschende Bild existiert nicht.

Folgendes Beispiel bietet eine einfache Möglichkeit EMF-Seiten für die Anzeige in einer Druckvorschau aus dem aktuellen Inhalt eines RtfEdit-Objekts zu generieren.

```
sub MakeEmfPages( aRtfEdit : handle;           // Deskriptor des RtfEdit  aPath      : alpha(250); // P
```

Beim RtfEdit-Objekt muss die Eigenschaft PrtDevice gesetzt sein, bevor der Befehl durchgeführt wird. Im anderen Fall wird der Laufzeitfehler \_ErrHdlInvalid generiert.

Die erzeugten EMF-Dateien können vom PrtMetaPicture-Objekt angezeigt werden.

Mögliche Laufzeitfehler:

<u>_ErrValueInvalid</u>	In (int2) ist eine ungültige Option angegeben.
<u>_ErrHdlInvalid</u>	In (obj) wurde kein Deskriptor eines <u>RtfEdit</u> oder in dem angegebenen Objekt ist kein <u>PrintDevice</u> angegeben.

Konstanten für Rtf-Befehle

Konstanten für Rtf-Befehle

Siehe RTF-Befehle

- WinRtfLoadAscii
- WinRtfLoadAuto
- WinRtfLoadInsert
- WinRtfLoadMix
- WinRtfLoadOem
- WinRtfLoadRtf
- WinRtfPicModeAuto
- WinRtfPicModeQuality
- WinRtfPicModeSpeed
- WinRtfSaveAscii
- WinRtfSaveAuto
- WinRtfSaveMark
- WinRtfSaveMix
- WinRtfSaveOem
- WinRtfSaveRtf
- WinRtfSearchCase
- WinRtfSearchDelete
- WinRtfSearchReplace
- WinRtfSearchUp
- WinRtfSearchWord
- WinRtfTabCenter
- WinRtfTabDecimal
- WinRtfTabLeft
- WinRtfTabNone
- WinRtfTabRight
- WinStreamBufField
- WinStreamBufText
- WinStreamCaption
- WinStreamNameBin
- WinStreamNameFile
- WinStreamNameText

**\_WinRtfLoadAscii**

**Text im ASCII-Format laden**

**Wert 2 / 0x0002**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfLoadBin()**

**Der Text wird im ASCII-Format geladen.**

**\_WinRtfLoadAuto**

**Format automatisch bestimmen**

**Wert 0 / 0x0000**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfLoadBin()**

**Das Format des zu ladenden Textes wird automatisch bestimmt. Handelt es sich nicht um einen RTF-Text, wird der Text im ASCII-Format gelesen.**



**\_WinRtfLoadInsert**

**Text einfügen**

**Wert 16 / 0x0010**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfLoadBin()**

**Der zu ladende Text wird in einen bestehenden Text eingefügt.**

**\_WinRtfLoadMix**

**Text mit Daten beim Laden mischen**

**Wert 8 / 0x0008**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfLoadBin()**

**Beim Laden des Textes werden die enthaltenen Platzhalter durch den entsprechenden Inhalt ersetzt. Die Platzhalter sind mit Markierungszeichen geklammert, die in der Eigenschaft RtfMixMarker des \_App-Objektes definiert werden kann.**

**Weitere Hinweise befinden sich im Abschnitt Text und Daten mischen.**

**\_WinRtfLoadOem**

**Text im OEM-Format laden**

**Wert 4 / 0x0004**

**Siehe WinRtfLoad(),**

**WinRtfLoadName()**

**Der Text wird im OEM-Format geladen.**

**\_WinRtfLoadRtf**

**Text im RTF-Format laden**

**Wert 1 / 0x0001**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfLoadBin()**

**Der Text wird im RTF-Format geladen.**

## Kontakt

**\_WinRtfPicModeAuto**

Qualität des Bildimports abhängig von der Farbtiefe und Größe des Bildes Wert 0

**WinRtfPicInsertMem()**,

Siehe **WinRtfPicInsertName()**,

**WinRtfPicModeSpeed**,

**WinRtfPicModeQuality**

Option bei **WinRtfPicInsertMem()** und **WinRtfPicInsertName()**. Beim Einfügen eines Bildes in ein **RtfEdit**-Objekt wird abhängig von der Größe und der Farbtiefe des Bildes automatisch entweder **WinRtfPicModeSpeed** oder **WinRtfPicModeQuality** verwendet.

**\_WinRtfPicModeQuality**

**Bildimport auf Qualität optimiert**

**Wert 2**

**WinRtfPicInsertMem(),**

Siehe **WinRtfPicInsertName(),**

**WinRtfPicModeSpeed,**

**WinRtfPicModeAuto**

**Option bei WinRtfPicInsertMem() und WinRtfPicInsertName(). Das Einfügen großer Bilder kann sehr lange dauern. Dafür bleibt die Qualität des Bildes auch beim Ändern der Größe im RtfEdit-Objekt erhalten. Wird ein Rtf-Text mit dieser Option gespeichert, wird die resultierende Datei größer als mit der Option WinRtfPicModeSpeed.**

**\_WinRtfPicModeSpeed**

**Bildimport auf Performance optimiert**

**Wert 1**

**WinRtfPicInsertMem()**,

Siehe **WinRtfPicInsertName()**,

**WinRtfPicModeQuality**,

**WinRtfPicModeAuto**

Option bei **WinRtfPicInsertMem()** und **WinRtfPicInsertName()**. Das Einfügen des Bildes in ein **RtfEdit**-Objekt ist performant, auch für große Bilder. Insbesondere beim Verkleinern des Bildes kann es jedoch zu Verlusten bei der Darstellung kommen. Beim Speichern des Rtf-Textes ist die resultierende Datei kleiner als mit der Option **WinRtfPicModeQuality**.

**\_WinRtfSaveAscii**

**Speichern im ASCII-Format**

**Wert 2 / 0x0002**

**WinRtfSave(),**

**Siehe WinRtfSaveName(),**

**WinRtfSaveBin()**

**Der Text wird im ASCII-Format gespeichert. Die Formatierungen gehen dabei verloren.**



**\_WinRtfSaveAuto**

**Format automatisch bestimmen**

**Wert 0 / 0x0000**

**WinRtfSave(),**

**Siehe WinRtfSaveName(),**

**WinRtfSaveBin()**

**Das Format wird automatisch bestimmt. In der Regel wird der Text im RTF-Format gespeichert.**

**\_WinRtfSaveMark**

**Markierung speichern**

**Wert 16 / 0x0010**

**WinRtfSave(),**

**Siehe WinRtfSaveName(),**

**WinRtfSaveBin()**

**Der markierte Bereich wird gespeichert.**

## Kontakt

**\_WinRtfSaveMix**

**Text mit Daten beim Laden mischen**

**Wert 8 / 0x0008**

**WinRtfSave(),**

**Siehe WinRtfSaveName(),**

**WinRtfSaveBin()**

**Beim Speichern des Textes werden die enthaltenen Platzhalter durch den entsprechenden Inhalt ersetzt. Die Platzhalter sind mit Markierungszeichen geklammert, die in der Eigenschaft RtfMixMarker des \_App-Objektes definiert werden können.**

**Weitere Hinweise befinden sich im Abschnitt Text und Daten mischen.**

**\_WinRtfSaveOEM**

**Speichern im OEM-Format**

**Wert 4 / 0x0004**

**Siehe** **WinRtfSave()**,

**WinRtfSaveName()**

**Der Text wird im OEM-Format gespeichert. Die Formatierungen gehen dabei verloren.**

**\_WinRtfSaveRtf**

**Speichern im RTF-Format**

**Wert 1 / 0x0001**

**WinRtfSave(),**

**Siehe WinRtfSaveName(),**

**WinRtfSaveBin()**

**Der Text wird im RTF-Format gespeichert.**

## Kontakt

**\_WinRtfSearchCase**

**Option beim Durchsuchen eines RTF-Textes Wert 2  
/ 0x0002**

**Siehe WinRtfSearch()**

**Option beim Befehl WinRtfSearch().**

**Mit dieser Option wird die Groß- und Kleinschreibung im Suchtext berücksichtigt. Der Suchtext muss in der gleichen Schreibweise im Text vorkommen.**

## Kontakt

**\_WinRtfSearchDelete**

**Option beim Durchsuchen eines RTF-Textes Wert**

**16 / 0x0010**

**Siehe WinRtfSearch()**

**Option beim Befehl WinRtfSearch(). Bei der Angabe dieser Option wird der in Range angegebene Text-Bereich ausgeschnitten.**

## Kontakt

**\_WinRtfSearchReplace**

**Option beim Durchsuchen eines RTF-Textes Wert 8**

**/ 0x0008**

**Siehe WinRtfSearch()**

**Option beim Befehl WinRtfSearch(). Bei der Angabe dieser Option wird der Suchtext durch einen anderen anzugebenden Text ersetzt. Der entsprechende Text wird beim Befehl WinRtfSearch() angegeben.**



## Kontakt

**\_WinRtfSearchUp**

**Option beim Durchsuchen eines RTF-Textes Wert 1**

**/ 0x0001**

**Siehe WinRtfSearch()**

**Option beim Befehl WinRtfSearch().**

**Mit dieser Option wird die Suchrichtung bestimmt. Wird die Option nicht angegeben, wird der Text vom Beginn zum Ende durchsucht. Mit der Option erfolgt die Suche vom Ende des Textes zum Anfang.**

## Kontakt

**\_WinRtfSearchWord**

**Option beim Durchsuchen eines RTF-Textes Wert 4  
/ 0x0004**

**Siehe WinRtfSearch()**

**Option beim Befehl WinRtfSearch().**

**Mit dieser Option wird nur nach ganzen Wörtern gesucht. Wörter werden durch Leerzeichen, Satzzeichen oder Bindestriche und Ähnlichem voneinander getrennt.**

## Kontakt

**\_WinRtfTabCenter**

zentrierten Tabulator setzen

Wert 2

Siehe **RtfTabMake()**

Es wird ein zentrierter Tabulator angelegt. Der Text wird zum Tabulator zentriert dargestellt.

## Kontakt

**\_WinRtfTabDecimal**

**Dezimal-Tabulator setzen**

**Wert 4**

**Siehe RtfTabMake()**

**Es wird ein Dezimal-Tabulator angelegt. Der Text wird zum Dezimalpunkt (kein Komma) zentriert.**

## Kontakt

**\_WinRtfTabLeft**

linksbündigen Tabulator setzen

Wert 1

Siehe **RtfTabMake()**

Es wird ein linksbündiger Tabulator angelegt. Der Text wird linksbündig zum Tabulator dargestellt.

## Kontakt

**\_WinRtfTabNone**

**bestehenden Tabulator löschen**

**Wert 0**

**Siehe RtfTabMake()**

**Der bestehende Tabulator an der angegebenen Position wird gelöscht.**

**\_WinRtfTabRight**

rechtsbündigen Tabulator setzen

Wert 3

Siehe **RtfTabMake()**

Es wird ein rechtsbündiger Tabulator angelegt. Der Text wird rechtsbündig zum Tabulator dargestellt.

**\_WinStreamBufField**

**Quelle des Textes**

**Wert 2**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamBufField gesetzt, wird der Inhalt des in DbFieldName angegebenen Datenbankfeldes angezeigt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad() bzw. WinRtfSave() anzugeben.**



**\_WinStreamBufText**

Quelle des Textes

Wert 3

Siehe WinRtfLoad(),

Die Konstante wird verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad() bzw. WinRtfSave() anzugeben.

## Kontakt

**\_WinStreamCaption**

**Quelle des Textes**

**Wert 1**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamCaption gesetzt, wird der in der Eigenschaft Caption vorhandene Text dargestellt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad() / WinRtfLoadName() bzw. WinRtfSave() / WinRtfSaveName() anzugeben.**

**\_WinStreamNameBin**

**Quelle des Textes**

**Wert 6**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamNameBin gesetzt, wird das in der Eigenschaft FileName binäre Objekt dargestellt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad() / WinRtfLoadName() bzw. WinRtfSave() / WinRtfSaveName() anzugeben.**

**\_WinStreamNameFile**

**Quelle des Textes**

**Wert 5**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamNameFile gesetzt, wird der Inhalt der in der Eigenschaft FileName angegebenen externen Datei dargestellt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad(), WinRtfLoadName(), WinRtfSave(), WinRtfSaveName(), WinDocLoadName() und WinDocSaveName() anzugeben.**

**\_WinStreamNameText**

**Quelle des Textes**

**Wert 4**

**StreamSource,**

**WinRtfLoad(),**

**Siehe WinRtfLoadName(),**

**WinRtfSave(),**

**WinRtfSaveName()**

**Wird die Eigenschaft StreamSource auf den Wert \_WinStreamNameText gesetzt, wird der in der Eigenschaft FileName angegebene interne Text dargestellt.**

**Die Konstante wird ebenfalls verwendet, um die Quelle bzw. das Ziel des Textes bei den Befehlen WinRtfLoad(), WinRtfLoadName(), WinRtfSave(), WinRtfSaveName(), WinDocLoadName() und WinDocSaveName() anzugeben.**

## Theme-Befehle

Befehle zum Bearbeiten eines Theme-Objekts

**Siehe** Verwandte  
Befehle, Theme,  
Befehlsgruppen,  
Befehlsliste

## Befehle

- WinThemeClose
- WinThemeDelete
- WinThemeOpen
- WinThemeSetDelete
- WinThemeSetOpen

obj ->

WinThemeClose([int1[,



alpha2]]) : int

Theme schließen

obj      Deskriptor des Theme-Objektes

int1      Optionen (optional)

alpha2    Neuer Name (optional)

Fehlerwert

ErrOk

Befehl wurde  
erfolgreich  
durchgeführt.

ErrNameInvalid Der Name

Resultat int      entspricht  
nicht den  
gültigen



-70

Konventionen.  
Theme-Objekt  
(obj) nicht  
gesperrt.

Siehe Verwandte Befehle, Theme

Der Befehl entsperrt das Theme-Objekt, sofern dies durch die Option

WinThemeOpenLock zuvor gesperrt wurde, und schließt es. Der Deskriptor ist nach dem Aufruf nicht mehr gültig.

Im Parameter (obj) wird der Deskriptor auf ein, mit WinThemeOpen() geöffnetes Theme-Objekt angegeben.

Folgende Optionen können im Parameter (int1) angegeben werden:

- WinThemeCloseSave

Das Theme wird gespeichert. Alternativ kann im Argument (alpha2) auch ein abweichender Name angegeben werden. Dies ist auch dann notwendig, wenn ein vordefiniertes Theme geöffnet und geändert wurde und nun in der Datenbank gespeichert werden soll.

Im Argument (alpha2) kann ein Name für das Thema angegeben werden, wenn die Option WinThemeCloseSave verwendet wird. Falls bereits ein Theme unter dem Name existiert, wird es überschrieben.

Von der Funktion können folgende Fehlerwerte zurückgegeben werden:

ErrOk

Befehl wurde erfolgreich durchgeführt.

ErrNameInvalid Der Name entspricht nicht den gültigen Konventionen.

-70

Theme-Objekt (obj) nicht gesperrt.

Mögliche Laufzeitfehler

ErrHdlInvalid      Der Deskriptor (obj) ist ungültig.

ErrValueInvalid Es wurde eine ungültige Option (int1) angegeben.

WinThemeDelete(alpha1) :



int

Theme löschen

alpha1 Name des Themes

Fehlerwert

ErrOk

Befehl wurde  
erfolgreich  
durchgeführt.

ErrNameInvalid Der Name

(alpha1)  
entspricht  
nicht den  
gültigen

Resultat int



ErrLocked

Konventionen.  
Das Theme  
(alpha1) ist  
gesperrt.

ErrUnavailable

Das Theme  
(alpha1)  
existiert nicht.

Siehe Verwandte Befehle, Theme

Der Befehl löscht ein in der Datenbank gespeichertes Theme. Der Name des zu löschenden Themes wird in (alpha1) angegeben.

Folgende Fehlerwerte können von der Funktion zurückgegeben werden:

ErrOk

Befehl wurde erfolgreich durchgeführt.

ErrNameInvalid Der Name (alpha1) entspricht nicht den gültigen Konventionen.

ErrLocked

Das Theme (alpha1) ist gesperrt.

ErrUnavailable


Das Theme (alpha1) existiert nicht.



## Kontakt

WinThemeOpen([alpha1[,  
int2]]) : handle Theme öffnen



alpha1 Name des Themes (optional)  
Optionen (optional)  
0 Theme mit dem Namen (alpha1) öffnen  
WinThemeOpenLock Theme exklusiv sperren  
WinThemeOpenFirst Erstes Theme öffnen  
WinThemeOpenLast Letztes Theme öffnen  
int2 WinThemeOpenNext Nächstes Theme, ausgehend vom Namen (alpha1) öffnen  
WinThemeOpenPrev Vorhergehendes Theme, ausgehend vom Namen (alpha1) öffnen  
Deskriptor des Theme-Objektes oder Fehlerwert  
> 0 Deskriptor des Theme-Objektes  
ErrNameInvalid Es handelt sich nicht um ein vordefiniertes Theme oder der Theme-Name entspricht nicht  den gültigen Konventionen.  
Resultat handle  
ErrUnavailable Theme existiert nicht.  
ErrLocked Das Theme wurde bereits von einem anderen Client gesperrt.

Siehe Verwandte Befehle, Theme, Blog

Der Befehl öffnet ein vordefiniertes oder ein in der Datenbank gespeichertes Theme und gibt einen Deskriptor vom Typ HdlTheme zurück.

Der Name des zu ladenen Themes wird in (alpha1) angegeben. Es können Namen von vordefinierten oder in der Datenbank gespeicherten Themes angegeben werden:

## Kontakt

- Beginnt der Name mit einem Unterstrich \_, wird ein vordefiniertes Theme geöffnet. Es existieren drei vordefinierte Themes: '\_OfficeBlue', '\_OfficeDark' und '\_WindowsColor'.
- Ist der Name eine leere Zeichenkette, dann wird das Theme '\_OfficeBlue' geöffnet.
- Bei der Angabe anderer Namen, wird das Theme aus der Datenbank geladen.

Im Argument (int2) können folgende Optionen angegeben werden:

- 0

Das durch (alpha1) angegebene Theme wird geladen. Falls kein Theme mit diesem Name existiert, wird der Wert \_ErrUnavailable zurückgegeben.

- \_WinThemeOpenLock

Lädt das Theme und sperrt es exklusiv. Dies ist die Voraussetzung, damit Theme-Eigenschaften geändert werden können. Soll ein Theme nicht geändert, sondern nur gelesen werden, kann die Option entfallen.

- \_WinThemeOpenFirst

Lädt das erste Theme (aufsteigend nach Theme-Name) aus der Datenbank. Ist kein Theme in der Datenbank vorhanden, wird der Fehlerwert \_ErrUnavailable zurückgegeben.

- \_WinThemeOpenLast

Lädt das letzte Theme aus der Datenbank. Ist kein Theme in der Datenbank vorhanden, wird der Fehlerwert \_ErrUnavailable zurückgegeben.

- \_WinThemeOpenNext

Lädt ausgehend vom angegebenen Namen (alpha1) das nächste Theme. Ist kein nächstes Theme vorhanden, wird der Fehlerwert \_ErrUnavailable zurückgegeben.

- \_WinThemeOpenPrev

Lädt ausgehend vom angegebenen Namen (alpha1) das vorhergehende Theme. Ist kein vorhergehendes Theme vorhanden, wird der Fehlerwert \_ErrUnavailable zurückgegeben.

Die Optionen \_WinThemeOpenFirst, \_WinThemeOpenLast, \_WinThemeOpenNext und \_WinThemeOpenPrev können mit der Option \_WinThemeOpenLock kombiniert werden.



Ist eine Option außer 0 angegeben, bezieht sich diese auf ein Theme das in der Datenbank gespeichert ist. Die Optionen dürfen nicht für vordefinierte Themes angegeben werden.

Beim Erstellen eines Theme-Objektes wird die Eigenschaft ThemeBaseName auf das zugrundeliegende vordefinierte Theme gesetzt.

Außer einem Deskriptor auf ein Theme-Objekt können folgende Fehlerwerte von der Funktion zurückgegeben werden:

## Kontakt

ErrNameInvalid Es handelt sich nicht um ein vordefiniertes Theme oder der  
Theme-Name entspricht nicht den gültigen Konventionen.  
ErrUnavailable Theme existiert nicht.  
ErrLocked Das Theme wurde bereits von einem anderen Client gesperrt.  
Mögliche Laufzeitfehler

ErrValueInvalid In (int2) wurde eine unbekannte oder ungültige Option übergeben.

obj ->

WinThemeSetDelete(int1) :



int

ThemeSet löschen

obj Deskriptor des Themes

int1 Nummer des zu löschenden

ThemeSets

Optionen (optional)

0

ThemeSet  
mit der  
Nummer  
(int1)  
öffnen

int2

WinThemeSetOpenCreate Neues

ThemeSet  
mit der  
Nummer  
(int1)  
erstellen

Fehlerwert

ErrOk

Das  
ThemeSet  
wurde  
erfolgreich  
gelöscht.

Resultat int ErrUnavailable Das



ThemeSet  
mit der  
Nummer  
(int1)  
existiert  
nicht.

Siehe

Verwandte Befehle, Theme,

ThemeSet, WinThemeSetOpen()

Dieser Befehl löscht ein vorhandenes ThemeSet-Objekt eines Theme-Objektes (obj).

In (obj) muss ein Theme-Objekt angegeben werden (siehe WinThemeOpen()). Im Argument (int1) wird eine eindeutige Nummer eines ThemeSets innerhalb des Themes angegeben.

Folgende Fehlerwerte können von der Funktion zurückgegeben werden:

ErrOk

Das ThemeSet wurde erfolgreich gelöscht.

ErrUnavailable

Das ThemeSet mit der Nummer (int1) existiert nicht.



Damit das ThemeSet aus dem, in der Datenbank gespeicherten, Theme entfernt wird, muss das Theme mit WinThemeClose() und der Option WinThemeCloseSave gespeichert werden.

Mögliche Laufzeitfehler

## Kontakt

ErrHdlInvalid

Der Deskriptor (obj) ist ungültig oder kein Deskriptor eines  
Theme-Objektes.

ErrValueInvalid Die angegebene ThemeSet-Nummer (int1) ist  $\leq 0$ .

## Kontakt

obj -> WinThemeSetOpen(int1[,  
int2]) : handle  
ThemeSet öffnen



obj	<u>Deskriptor des Themes</u>	
	Nummer des zu erstellenden oder zu öffnenden	
int1	<u>ThemeSets</u>	
	Optionen (optional)	
	0	ThemeSet mit der
		Nummer (int1) öffnen
int2	<u>WinThemeSetOpenCreate</u>	Neues ThemeSet mit
		der Nummer (int1)
		erstellen
	Deskriptor des ThemeSets oder	
	Fehlerwert	
	> 0	Deskriptor des ThemeSets
	<u>ErrUnavailable</u>	Das ThemeSet mit der
Resultat <u>handle</u>		Nummer (int1) existiert
		nicht.
	<u>ErrExists</u>	Das ThemeSet mit der
		Nummer (int1) existiert
		bereits (bei Angabe von
		<u>WinThemeSetOpenCreate</u> ).



Siehe Verwandte Befehle, Theme, ThemeSet,

WinThemeOpen(), WinThemeSetDelete()

Dieser Befehl erstellt ein neues oder öffnet ein vorhandenes ThemeSet-Objekt eines Theme-Objektes (obj) und gibt einen Deskriptor vom Typ HdlThemeSet zurück. Das ThemeSet enthält alle im Theme definierten Theme-Elemente und deren Eigenschaften.

In (obj) muss ein Theme-Objekt angegeben werden (siehe WinThemeOpen()). Im Argument (int1) wird eine eindeutige Nummer eines ThemeSets innerhalb des Themes angegeben.

Im Argument (int2) können folgende Optionen angegeben werden:

- 0

Das durch (int1) angegebene ThemeSet wird geladen. Falls kein ThemeSet mit dieser Nummer im Theme (obj) existiert, wird der Wert ErrUnavailable zurückgegeben.

- WinThemeSetOpenCreate

Das durch (int1) angegebene ThemeSet wird erstellt. Falls bereits ein ThemeSet mit dieser Nummer im Theme (obj) existiert, wird der Wert ErrExists zurückgegeben.

Bei der Erstellung eines ThemeSet mit der Option WinThemeSetOpenCreate enthalten die Eigenschaften zunächst Default-Werte. Dadurch wird definiert, dass die Eigenschaften des ThemeSets dieselben Werte enthalten, wie das Theme (obj) selber.

## Kontakt

Für die Default-Werte gibt es folgende Konstanten, die bei den Eigenschaften gesetzt oder gelesen werden können:

WinThemeIntNULL Integer-Eigenschaften

WinThemeColorNULL Farb-Eigenschaften

WinThemeFontNULL Font-Eigenschaften

Außer einem Deskriptor auf ein ThemeSet können folgende Fehlerwerte von der Funktion zurückgegeben werden:

ErrUnavailable Das ThemeSet mit der Nummer (int1) existiert nicht.

ErrExists Das ThemeSet mit der Nummer (int1) existiert bereits (bei Angabe von WinThemeSetOpenCreate).



Damit neue oder geänderte ThemeSets gespeichert werden, muss das jeweilige Theme mit WinThemeClose() und der Option WinThemeCloseSave gespeichert werden.

Beispiel:

```
// ThemeSet mit Nummer 1 anlegen und Füll- sowie Textfarbe von Button-Objekten setzenThemeSet1 #
```

Mögliche Laufzeitfehler

ErrHdlInvalid

Der Deskriptor (obj) ist ungültig oder kein Deskriptor eines Theme-Objektes.

ErrValueInvalid

Die angegebene ThemeSet-Nummer (int1) ist  $\leq 0$  oder in (int2) wurde eine unbekannte oder ungültige Option übergeben.

ErrMemExhausted

Das ThemeSet konnte nicht allokiert werden (bei Angabe von WinThemeSetOpenCreate in (int2)).

## Kontakt

Konstanten für Theme-Befehle

Konstanten für Theme-Befehle

Siehe Theme-Befehle

<u>WinThemeCloseSave</u>	<u>Theme</u> beim Schließen speichern
<u>WinThemeOpenFirst</u>	Erstes <u>Theme</u> öffnen
<u>WinThemeOpenLast</u>	Letztes <u>Theme</u> öffnen
<u>WinThemeOpenLock</u>	<u>Theme</u> exklusiv sperren
<u>WinThemeOpenNext</u>	Nächstes <u>Theme</u> öffnen
<u>WinThemeOpenPrev</u>	Vorheriges <u>Theme</u> öffnen
<u>WinThemeSetOpenCreate</u>	Neues <u>ThemeSet</u> erstellen



## Kontakt

**\_WinThemeCloseSave**

**Theme speichern**

**Wert 1**

**Siehe WinThemeClose()**

**Option bei WinThemeClose(), mit der das Theme beim Schließen gespeichert wird.**

## Kontakt

**\_WinThemeOpenFirst**

**Erstes Theme öffnen**

**Wert 16 / 0x0010**

**Siehe WinThemeOpen()**

**Option bei WinThemeOpen(), mit der das erste Theme geöffnet wird.**

## Kontakt

**\_WinThemeOpenLast**

**Letztes Theme öffnen**

**Wert 128 / 0x0080**

**Siehe WinThemeOpen()**

**Option bei WinThemeOpen(), mit der das letzte Theme geöffnet wird.**

## Kontakt

**\_WinThemeOpenLock**

**Theme beim Öffnen sperren**

**Wert 1 / 0x01**

**Siehe WinThemeOpen()**

**Option bei WinThemeOpen(), mit der das Theme beim Öffnen exklusiv gesperrt wird.**

**Dies ist die Voraussetzung, damit Theme-Eigenschaften geändert werden können.**

## Kontakt

**\_WinThemeOpenNext**

**Nächstes Theme öffnen**

**Wert 32 / 0x0020**

**Siehe WinThemeOpen()**

**Option bei WinThemeOpen(), mit der das nächste Theme geöffnet wird.**

## Kontakt

**\_WinThemeOpenPrev**

**Vorhergehendes Theme öffnen**

**Wert 64 / 0x0040**

**Siehe WinThemeOpen()**

**Option bei WinThemeOpen(), mit der das vorhergehende Theme geöffnet wird.**

## Kontakt

**\_WinThemeSetOpenCreate**

Neues ThemeSet erstellen

Wert 1 / 0x0001

Siehe WinThemeSetOpen()

Option bei WinThemeSetOpen(), mit der ein neues ThemeSet angelegt wird.

## TreeView-Befehle

Befehle zum Bearbeiten eines TreeView-Objekts

Verwandte  
Befehle,

Siehe TreeView,  
Befehlsgruppen,  
Befehlsliste

Befehle

- WinTreeNodeAdd
- WinTreeNodeRemove
- WinTreeNodeSearch



obj ->

WinTreeNodeAdd([alpha1[,  
alpha2[, int3]]) : handle



Knoten zu Baum hinzufügen  
obj Baum oder Knoten

alpha1 Knotenname  
(optional)

alpha2 Knotentitel (optional)  
Nachfolgeknoten

int3  
(optional)

Resultat handle Knoten-Objekt  
Verwandte Befehle,

Siehe

TreeNode, TreeView

Mit diesem Befehl wird einem Baum-Objekt oder einem Knoten-Objekt (obj) ein neuer Knoten hinzugefügt. In (int3) kann ein Knoten der gleichen Ebene angegeben werden, der neue Knoten wird dann vor diesem Knoten in den Baum eingefädelt. Wird dieser Parameter nicht angegeben, wird der Knoten als letzter Knoten des in (obj) angegebenen Objektes angehängt. Name (alpha1) (siehe Name) und Titel (alpha2) (siehe Caption) können direkt beim Aufruf bestimmt werden, alle weiteren Eigenschaften müssen mit Hilfe des Resultats gesetzt werden.

Als Resultat wird der Deskriptor des neuen Knoten zurückgeliefert. Im Fehlerfall ist das Ergebnis 0.

Beispiel:

```
// Fügt dem Baum $TreeView einen Knoten mit dem// Namen 'node1' und dem Titel 'Node 1' hinzuHdlN
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Baum oder Knoten (obj) ungültig

obj ->



**WinTreeNodeRemove([logic1])**

**Knoten aus Baum entfernen**

Baum /

obj

**Knoten**

**Knoten**

**logic1 leeren**

(optional)

**Verwandte**

Siehe Befehle,

**TreeNode**,

**TreeView**

Dieser Befehl entfernt ein Knoten-Objekt (obj) inklusive aller Unterknoten aus einem Baum-Objekt. Wird der optionale Parameter (logic1) auf true gesetzt, werden nur die Unterknoten des Knotens (obj) entfernt.

In (obj) kann auch ein Baum-Objekt angegeben werden. Es werden dann alle Knoten des Baums entfernt.

Ist die Eigenschaft AutoUpdate auf false gesetzt (siehe WinUpdate()), werden die Objekte erst beim Aktivieren des Updates entfernt.

**Beispiele:**

```
// Knoten tHdlNode entferntHdlNode->WinTreeNodeRemove();// Baum leerentHdlTree->WinTreeNodeRemo
```

## Kontakt

obj ->

WinTreeNodeSearch(alpha1[, int2[,  
handle3]]) : handle



Knoten im Baum suchen

obj Startobjekt der Suche: TreeView- oder  
TreeNode-Objekt

alpha1 Suchtext

Optionen (optional)

WinTreeNodeSearchCaption

Durchsucht die  
Caption-Eigenschaft

WinTreeNodeSearchCustom

Durchsucht die  
Custom-Eigenschaft

WinTreeNodeSearchCI

Ignoriert

WinTreeNodeSearchLike

Groß-/Kleinschreibung  
Wildcard-Vergleich

WinTreeNodeSearchLikeAuto

Wildcard-Vergleich mit  
automatischer

int2 WinTreeNodeSearchToken

Maskierung  
Begriffsorientierte

WinTreeNodeSearchRegEx

Suche  
Vergleich mit  
regulären Ausdrücken

WinTreeNodeSearchChildrenOnly Nur Kind-Knoten

WinTreeNodeSearchUp

durchsuchen  
Sucht nach oben

WinTreeNodeSearchNoSelect

Treffer nicht

WinTreeNodeSearchStart

selektieren  
Sucheingabefeld

öffnen

handle3 Referenzknoten für untergeordnete Suche

> 0 Deskriptor des gefundenen  
Knotens

Resultat handle

= 0 Kein Knoten gefunden

Siehe Verwandte Befehle, TreeNode, TreeView,

EvtNodeSearch, Blog

Mit diesem Befehl wird in einem TreeView-Objekt oder einem TreeNode-Objekt (obj) nach einem bestimmten TreeNode gesucht.



Ist ein Referenzknoten (handle3) angegeben und wird das Ende der Unterelemente des übergebenen TreeView bzw. TreeNode-Objektes (obj) erreicht, beginnt die Suche von vorne, bis zum wieder Erreichen des Referenzknotens.

Wird als Startobjekt (obj) ein TreeNode-Objekt übergeben, wird standardmäßig auch in höheren Baumebenen weiter gesucht.

Folgende Optionen (int2) können angegeben werden:

## Kontakt

- WinTreeNodeSearchCaption

Vergleicht den Suchtext mit Caption-Eigenschaft des TreeNode-Objektes.

- WinTreeNodeSearchCustom

Vergleicht den Suchtext mit Custom-Eigenschaft des TreeNode-Objektes.

- WinTreeNodeSearchCI

Die Groß-/Kleinschreibung wird bei der Suche ignoriert.

- WinTreeNodeSearchLike

Führt einen Wildcard-Vergleich durch.

- WinTreeNodeSearchLikeAuto

Führt einen Wildcard-Vergleich durch. Der Suchtext wird automatisch mit \*<Suchtext>\* maskiert. Dies entspricht dem Verhalten im Designer.

- WinTreeNodeSearchToken

Führt eine begriffsorientierte Suche durch.

- WinTreeNodeSearchRegEx Vergleich mittels  
regulären Ausdrücken.

- WinTreeNodeSearchChildrenOnly

Nur untergeordnete Knoten werden durchsucht.

- WinTreeNodeSearchUp

Die Suche wird nach oben durchgeführt.

- WinTreeNodeSearchNoSelect

Der gefundene Knoten wird nicht ausgewählt.

- WinTreeNodeSearchStart

Das Sucheingabefeld wird geöffnet. Hierbei wird das Sucheingabefeld mit dem Wert von (alpha1) vorbelegt. Für die Suche werden die restlichen angegebenen Optionen verwendet.



Die Prozedur bleibt beim Öffnen des Sucheingabefeldes nicht stehen. Dadurch ist der Rückgabewert der Funktion, bei Angabe dieser Option, immer 0.

Mehrere Optionen können miteinander kombiniert werden. Die Kombination erfolgt durch eine binäre ODER-Verknüpfung.



Die Optionen WinTreeNodeSearchLike, WinTreeNodeSearchLikeAuto, WinTreeNodeSearchToken und WinTreeNodeSearchRegEx können nicht miteinander kombiniert werden.

Ist weder WinTreeNodeSearchCaption noch WinTreeNodeSearchCustom gesetzt wird trotzdem das Ereignis EvtNodeSearch aufgerufen.

## Kontakt

Wird die Option (int2) WinTreeNodeSearchChildrenOnly angegeben, kann in (handle4) ein Referenzknoten angegeben werden. Dabei wird nur unterhalb des Knotens (obj) vom Referenzknoten (handle4) ausgehend gesucht. Ist bei der Option kein Referenzknoten (handle4) angegeben, wird ab dem ersten untergeordneten Knoten gesucht.

### Resultat

Wird ein TreeNode gefunden, ist das Resultat der Deskriptor des TreeNode-Objektes. Wurde kein TreeNode gefunden, ist das Resultat 0. Zusätzlich sind die folgenden Fehlerwerte möglich:

Fehlerwert	Bedeutung
<u>ErrUnavailable</u>	Der aktuell geprüfte Knoten wurde im Ereignis <u>EvtNodeSearch</u> gelöscht. Daher kann die Suche nicht fortgesetzt werden.
<u>ErrRegExRuleSyntax</u>	Syntaxfehler im regulären Ausdruck
<u>ErrRegExBadEscapeSequence</u>	Nicht aufgelöste Escape-Sequenz im Ausdruck
<u>ErrRegExPropertySyntax</u>	Ungültige Unicode-Eigenschaft
<u>ErrRegExNotSupported</u>	Verwendung einer Funktion, die nicht unterstützt wird
<u>ErrRegExMismatchedParentheses</u>	Falsch verschachtelte Klammern im regulären Ausdruck
<u>ErrRegExNumberTooBig</u>	Dezimalzahl zu groß
<u>ErrRegExBadInterval</u>	Fehler im {min,max} Intervall
<u>ErrRegExMaxLtMin</u>	Im Intervall {min,max} ist max kleiner als min
<u>ErrRegExInvalidBackRef</u>	Rückbezug auf eine nicht vorhandene Referenz
<u>ErrRegExInvalidFlag</u>	Ungültiger Modus
<u>ErrRegExLookBehindLimit</u>	Rückschau Ausdrücke müssen eine beschränkte maximale Länge haben
<u>ErrRegExSetContainsString</u>	Reguläre Ausdrücke können keine UnicodeSets mit Zeichenketten beinhalten
<u>ErrRegExMissingCloseBracket</u>	Fehlende schließende Klammer in einem Klammerausdruck
<u>ErrRegExInvalidRange</u>	In einer Zeichenmenge [x-y] ist x größer als y
<u>ErrRegExStackOverflow</u>	Stapelüberlauf in der Ablaufverfolgung des regulären Ausdrucks

Die Suche wird so lange durchgeführt, bis ein TreeNode gefunden, die Suche im Ereignis EvtNodeSearch mit dem Rückgabewert false abgebrochen oder das Startobjekt (obj) bzw. der Referenzknoten (handle4) bei WinTreeNodeSearchChildrenOnly wieder erreicht wurde.

### Beispiel:

```
// Findet alle untergeordneten TreeNodes, die mit '.doc' enden und ändert die Schriftartfor{ ttfi
```

### Mögliche Laufzeitfehler:

## Kontakt

In (obj) oder (handle3) wurde kein gültiger Deskriptor eines TreeView- oder TreeNode-Objektes übergeben.

ErrHdlInvalid

In (int2) ist WinTreeNodeSearchChildrenOnly gesetzt und (handle3) ist kein untergeordnetes TreeNode-Objekt von (obj).

ErrValueInvalid Es wurde eine ungültige Kombination an Optionen (int2) übergeben.

## Konstanten für TreeView-Befehle

Liste der Konstanten für TreeView-Befehle

Siehe [TreeView-Befehle](#)

Konstanten

- [WinTreeNodeSearchCaption](#)
- [WinTreeNodeSearchChildrenOnly](#)
- [WinTreeNodeSearchCI](#)
- [WinTreeNodeSearchCustom](#)
- [WinTreeNodeSearchLike](#)
- [WinTreeNodeSearchLikeAuto](#)
- [WinTreeNodeSearchNoSelect](#)
- [WinTreeNodeSearchRegEx](#)
- [WinTreeNodeSearchStart](#)
- [WinTreeNodeSearchToken](#)
- [WinTreeNodeSearchUp](#)

## Kontakt

\_WinTreeNodeSearchCaption Caption-Eigenschaft der TreeNode-Objekte durchsuchen Wert 1 / 0x0001

Siehe WinTreeNodeSearch(),

Option bei WinTreeNodeSearch() und in den SearchFlags, mit der die Caption-Eigenschaft der TreeNode-Objekte durchsucht wird.



## Kontakt

**\_WinTreeNodeSearchChildrenOnly**

Suche nur nach untergeordneten TreeNode-Objekten Wert 8 /  
0x0008

Siehe WinTreeNodeSearch(),

Option bei WinTreeNodeSearch() und in den SearchFlags, mit der nur nach untergeordneten TreeNode-Objekten gesucht wird.

## Kontakt

**\_WinTreeNodeSearchCI Groß-/Kleinschreibung bei der Suche ignorieren Wert 4 / 0x0004**

**Siehe** WinTreeNodeSearch(),

**Option bei WinTreeNodeSearch() und in den SearchFlags, mit der bei der Suche die Groß-/Kleinschreibung ignoriert wird.**

## Kontakt

\_WinTreeNodeSearchCustom Custom-Eigenschaft der TreeNode-Objekte durchsuchen Wert 2 / 0x0002

Siehe WinTreeNodeSearch(),

Option bei WinTreeNodeSearch() und in den SearchFlags, mit der die Custom-Eigenschaft der TreeNode-Objekte durchsucht wird.

## Kontakt

**\_WinTreeNodeSearchLike**

Suche mit Wildcard-Vergleich durchführen

Wert 32 / 0x0020

Siehe WinTreeNodeSearch(),

Option bei WinTreeNodeSearch() und in den SearchFlags, mit der ein Wildcard-Vergleich bei der Suche durchgeführt wird.

## Kontakt

**\_WinTreeNodeSearchLikeAuto**

Suche mit Wildcard-Vergleich und automatischer Maskierung durchführen Wert 16 / 0x0010

Siehe WinTreeNodeSearch(),

Option bei WinTreeNodeSearch() und in den SearchFlags, mit der ein Wildcard-Vergleich bei der Suche durchgeführt wird. Der Suchbegriff wird automatisch mit \*<Suchbegriff>\* maskiert.

## Kontakt

**\_WinTreeNodeSearchNoSelect**

**Suchtreffer nicht selektieren**

**Wert 8192 / 0x2000**

**Siehe WinTreeNodeSearch()**

**Option bei WinTreeNodeSearch(), mit der der Treffer bei der Suche nicht ausgewählt wird.**

## Kontakt

**\_WinTreeNodeSearchRegEx**

Suche mit regulären Ausdrücken durchführen Wert

128 / 0x0080

Siehe WinTreeNodeSearch(),

Option bei WinTreeNodeSearch() und in den SearchFlags, mit der die Suche mit regulären Ausdrücken durchgeführt wird.

## Kontakt

**\_WinTreeNodeSearchStart**

**Sucheingabefeld öffnen**

**Wert 16384 / 0x4000**

**Siehe WinTreeNodeSearch()**

**Option bei WinTreeNodeSearch(), mit der das Sucheingabefeld geöffnet wird.**



**Die Prozedur bleibt beim Öffnen des Sucheingabefeldes nicht stehen. Dadurch ist der Rückgabewert der Funktion, bei Angabe dieser Option, immer 0.**



## Kontakt

**\_WinTreeNodeSearchToken**

**Begriffsorientierte Suche durchführen**

**Wert 64 / 0x0040**

**Siehe** WinTreeNodeSearch(),

**Option bei WinTreeNodeSearch() und in den SearchFlags, mit der eine begriffsorientierte Suche durchgeführt wird.**

## Kontakt

**\_WinTreeNodeSearchUp**

**Suche nach oben durchführen**

**Wert 4096 / 0x1000**

**Siehe WinTreeNodeSearch()**

**Option bei WinTreeNodeSearch(), mit der die Suche vom angegebenen Startobjekt nach oben durchgeführt wird.**

## Kontakt

### Fokusbefehle

Befehle zum Ermitteln/Setzen des Eingabefokus

Verwandte  
Befehle,

**Siehe** Befehlsgruppen,  
Befehlsliste

### Befehle

- WinFocusGet
- WinFocusSet

### Konstanten

- StandardClient

WinFocusGet() : int         Objekt mit

Eingabefokus ermitteln

Resultat int Objekt mit Eingabefokus

Siehe Verwandte Befehle, TabPos

Es wird das Objekt zurückgegeben, das den Eingabefokus besitzt. 0 wird dann zurückgegeben, wenn das Objekt nicht ermittelt werden kann. Dies ist zum Beispiel dann der Fall, wenn zum Zeitpunkt der Befehlsausführung eine andere Applikation aktiv ist.

**Beispiel:**

```
// Objekt mit Eingabefokus ermittelntHdlObj # WinFocusGet();
```

## Kontakt

obj -> WinFocusSet([logic1]) :



int

Objekt mit Eingabefokus setzen

obj        Zu fokussierendes Objekt

logic1    Ereignisaktivierung (optional)

Resultat int Momentan fokussiertes Objekt

Siehe      Verwandte Befehle, TabPos

Dem Befehl wird das Objekt übergeben, auf den der Fokus gesetzt werden soll. Der Rückgabewert dieses Befehls ist entweder der Deskriptor des vorhergehenden Objekts oder der Wert 0. Ist der Wert 0, kann das unterschiedliche Ursachen haben. Zum einen kann das Objekt als nicht Visible oder als Disabled definiert worden sein, zum anderen kann das Objekt der Standardclient sein oder einem anderen Prozess gehören.

Durch die Übergabe der Konstanten \_StandardClient kann der Fokus auf den textbasierten Standardclient gesetzt werden.

Es ist zu beachten, dass ein Objekt nicht den Fokus erlangen kann, wenn es nicht sichtbar (Visible = false) oder gesperrt (Disabled = true) ist. Unterstützt ein Objekt keine Eingabe (zum Beispiel das Objekt Label), kann es ebenfalls keinen Fokus bekommen.

Mit dem optionalen Parameter (logic1) können während der Verarbeitung eines Ereignisses weitere Ereignisse unterdrückt (logic1 = false oder Parameter nicht angeben) werden. Wird in einer Funktion, die durch EvtFocusTerm aufgerufen wurde, der Fokus neu gesetzt, ist das nächste Ereignis das EvtFocusInit des neuen Eingabeobjekts.

Beispiel:

```
// Fokus auf Objekt '$edClItNo' setzen $edClItNo->WinFocusSet();
```



Der Befehl kann ebenfalls in dem Ereignis EvtInit eines Frame-Objekts aufgerufen bzw. zwischen dem Laden des Dialogs mit WinOpen() und dem Anzeigen mit WinDialogRun() durchgeführt werden. Da zu diesem Zeitpunkt ein Fokus nicht zugewiesen werden kann, erfolgt die Ausführung des Befehls verzögert. Der Rückgabewert des Befehls ist dann immer 0.

Mögliche Laufzeitfehler:

\_ErrHdlInvalid Objekt (obj) ungültig

## Kontakt

**\_StandardClient**

**Konstante für den Standardclient**

**Wert 10003**

**Siehe WinFocusSet()**

**Mit dieser Konstante kann der Fokus auf den Standardclient gesetzt werden.**

**Der Befehl WinFocusGet() liefert nicht diese Konstante zurück. Es kann also nicht ermittelt werden, ob der Standardclient den Fokus besitzt.**

## Ereignisbefehle

### Verändern der Ereignisse

Liste sortiert

nach

Gruppen,

Siehe

Alphabetische

Liste aller

### Befehle

- ComEvtProcessGet
- ComEvtProcessSet
- WinEvtProcessGet
- WinEvtProcessSet
- WinEvtProcNameGet
- WinEvtProcNameSet