

RecDeleteAll(int1)



Alle Datensätze löschen

int1 Dateinummer

Verwandte

Siehe Befehle,

RecDelete()

Diese Funktion löscht den kompletten Inhalt der angegebenen Datei (int1). Diese Anweisung kann auch im Mehrbenutzerbetrieb eingesetzt werden. Dabei werden allerdings auch alle evtl. gesperrten Datensätze in der angegebenen Datei gelöscht. Je nach Größe der Datei benötigt RecDeleteAll() eine gewisse Zeit zum Löschen.

Mögliche Laufzeitfehler:

ErrNoFile Datei nicht vorhanden

RecInfo(int1, int2[, bigint3]) :
bigint



Datensatzinformationen ermitteln

int1	Dateinummer oder Datensatzpuffer-Deskriptor Informationstyp	
	<u>RecCount</u>	Datensatzanzahl ermitteln
	<u>RecID</u>	Datensatz-ID ermitteln
	<u>RecSetID</u>	Datensatz-ID setzen
	<u>RecLen</u>	Datensatzgröße ermitteln
	<u>RecLenPacked</u>	Gepackte Datensatzgröße ermitteln
int2	<u>RecLockedBy</u>	Sperrenden Benutzer ermitteln
	<u>RecGetPos</u>	Schlüsselposition ermitteln
	<u>RecGetPosReverse</u>	Schlüsselposition ermitteln
	<u>RecGetPrime</u>	Prime-Counter der Datei ermitteln
	<u>RecUpdateCounter</u>	Update-Zähler der Datei ermitteln

Schlüsselnummer /
bigint3 Selektions-Deskriptor / Datensatz-ID
(optional)

Resultat bigint Datensatzinformation

Siehe Verwandte Befehle, RecBufCreate()


In (int1) wird die Nummer der Datei angegeben. Bei Verwendung der Optionen RecID und RecSetID kann in (int1) auch der Deskriptor eines Datensatzpuffers angegeben werden. In (int2) wird der Informationstyp festgelegt. Folgende Informationen können abgerufen werden:

RecCount

Mit dieser Option wird die Anzahl der Datensätze in der Datei ermittelt. Dabei kann in (bigint3) ein Selektionsdeskriptor angegeben werden, wodurch die Anzahl der Sätze in der Hauptergebnismenge der Selektion ermittelt wird. Die Anzahl der Sätze in einer verknüpften Ergebnismenge wird mit der Funktion RecLinkInfo() ermittelt.


RecID

Das Resultat ist die interne ID des aktuell geladenen Satzes der Datei. Soll die interne ID aus einem Datensatzpuffer ermittelt werden, muss in (int1) der Deskriptor des Datensatz-Puffers angegeben werden. Das Resultat ist ungültig, wenn kein Satz aus der Datei geladen ist.

 Bei der Datensatz-ID handelt es sich um eine interne ID, die sich durch bestimmte Aktionen (z. B. Export und Import der Datensätze) ändern kann. Sie sollte daher nicht als dauerhafte Referenz auf einen Datensatz verwendet werden.

RecSetID

Die Datensatz-ID wird auf den in (bigint3) übergebenen Wert gesetzt. Das Zurückschreiben eines Datensatzes mit dem Befehl RecReplace() erfolgt über die Datensatz-ID. Durch eine Änderung der Datensatz-ID kann der ursprünglich gelesene Datensatz nicht mehr zurückgeschrieben werden. Soll die interne ID eines Datensatzpuffers gesetzt werden, muss in (int1) der Deskriptor des Datensatzpuffers angegeben werden.

 Ist die in (bigint3) übergebene Datensatz-ID kleiner als 0, wird der Laufzeitfehler ErrValueInvalid ausgelöst.

RecLen

Die effektive Länge des im Hauptspeicher befindlichen Datensatzes in der Datei wird ermittelt. Das Resultat gibt die Größe des Datensatzes in Bytes zurück. In der effektiven Länge sind alphanumerische Felder nur mit ihrer aktuellen Länge berücksichtigt.

RecLenPacked


Die gepackte Länge des zuletzt gelesenen (z. B. RecRead()) oder geschriebenen (z. B. RecInsert()) Datensatzes in der Datei wird ermittelt. Änderungen am Feldpuffer werden nicht berücksichtigt. Das Resultat gibt die gepackte Größe des Datensatzes in Bytes zurück.

RecLockedBy

Wird bei einem Zugriff in die Datenbank auf einen gesperrten Datensatz zugegriffen, so kann hiermit die ID des Benutzers ermittelt werden, der den Datensatz gesperrt hat. Gleiches gilt auch beim Zugriff auf Texte, Selektionen oder Listenformate. Dabei ist die Dateinummer in (int1) ohne Bedeutung.

RecGetPos

In Dialogen ist oftmals die Position des aktuellen Satzes innerhalb der Datei (nach Schlüsselsortierung) wesentlich.

 Die Schlüsselposition eines Satzes gibt allerdings nur die ungefähre Position an, da eine exakte Positionsermittlung zu lange dauern würde (bedingt durch variable Schlüssellängen und dynamisches Speichermanagement in der Datenbank).

Die Nummer des Schlüssels oder eines Selektionsdeskriptors wird in (bigint3) übergeben.

Ausgehend von einer Datei mit 100 Datensätzen, die alle hintereinander durchnummeriert sind und der Datensatz mit der Nummer 4 geladen wurde, wird als Ergebnis von RecInfo(..., _RecGetPos) 4 zurückgegeben.

_RecGetPosReverse

Analog zu _RecGetPos, die Position des aktuellen Satzes innerhalb der Datei wird jedoch nach umgekehrter Schlüsselsortierung ermittelt.



Die Schlüsselposition eines Satzes gibt allerdings nur die ungefähre Position an, da eine exakte Positionsermittlung zu lange dauern würde (bedingt durch variable Schlüssellängen und dynamisches Speichermanagement in der Datenbank).

Ausgehend von einer Datei mit 100 Datensätze, die alle hintereinander durchnummeriert sind und der Datensatz mit der Nummer 4 geladen wurde, wird als Ergebnis von RecInfo(..., _RecGetPosReverse) 97 zurückgegeben.

_RecGetPrime

Das Resultat ist der Prime-Counter der Datei. Dieser Wert wird jedes Mal verändert, wenn ein Datensatz hinzugefügt wird. Wird ein Datensatz gelöscht, verändert sich der Wert nicht.

_RecUpdateCounter

Das Resultat ist der Update-Zähler der Datei. Dieser Wert wird jedes Mal verändert, wenn ein Datensatz erstellt, verändert oder gelöscht wird.

_RecModified


Das Resultat ist der Zeitstempel der letzten Änderung des Datensatzes.

Mögliche Laufzeitfehler:

<u>_ErrNoFile</u>	Die Datei (int1) ist nicht vorhanden.
<u>_ErrNoKey</u>	Der Schlüssel (bigint3) ist nicht vorhanden (bei <u>_RecGetPos</u>).
<u>_ErrHdlInvalid</u>	Der Selektions-Deskriptor (bigint3) (bei <u>_RecCount</u>) oder Datensatzpuffer-Deskriptor (int1) ist ungültig.
<u>_ErrFileInvalid</u>	Die Dateinummer (int1) passt nicht zu der Selektion (bigint3).
<u>_ErrValueInvalid</u>	Unter der Verwendung der Option (int2) <u>_RecSetID</u> wurde als Wert (bigint3) ein Wert kleiner als 0 angegeben.

RecInsert(int1, int2) : int
Datensatz einfügen



int1 Dateinummer oder Deskriptor eines
 Datensatzpuffers
 Optionen
 _RecLock Datensatz sperren
 _RecSingleLock Datensatz einfach
 sperren
int2 _RecSharedLock Datensatz mehrfach
 sperren
 _RecEarlyCommit Änderung trotz
 Transaktion
 durchführen
 Einfügeresultat
 _rOk Einfügen erfolgreich
 _rExists Datensatz existiert
 bereits
 _rNoRights Benutzerrechte
 nicht ausreichend
 _rDeadlock Verklemmung
 aufgetreten
Resultat int _rFailed Zugriff auf 
 temporären Baum
 nicht möglich
 (Clients < 5.8.11
 erhalten _rNoRec).
 _ErrDbComm Verbindungsabbruch
 zu einer
 verbundenen
 Datenbank
 aufgetreten.

Siehe Verwandte Befehle, RecDelete(),
RecReplace(), RecBufCreate(),
Datensatzpuffer

Mit dieser Funktion werden die momentan im Speicher befindlichen Feldpuffer einer Datei (bestehend aus den aktuellen Feldinhalten) als neuer Satz in eine Datei eingefügt. Wird in (int1) die Dateinummer angegeben, wird der Inhalt der Feldpuffer der Datei eingefügt. Es kann auch der Deskriptor eines mit RecBufCreate() erzeugten Datensatzpuffers angegeben werden. Dann wird der Inhalt des übergebenen Datensatzpuffers in die dazugehörige Datei übertragen. Als Option (int2) kann angegeben werden, ob der Satz nach dem Einfügen gesperrt sein soll (_RecLock, _RecSingleLock oder _RecSharedLock).

Es ist bei dieser Funktion darauf zu achten, dass die Felder des einzufügenden Datensatzes den gewünschten Inhalt haben, insbesondere im Hinblick auf die sich ergebenden Schlüsselwerte.

Kontakt

Bei der Angabe von RecEarlyCommit können innerhalb einer Transaktion (siehe DtaBegin()) Änderungen am Datenbestand vorgenommen werden, ohne dass es zu Wartezuständen bei anderen Transaktionen kommt.



Wird die Funktion durch den Evaluierungs-Client ausgeführt, kann als Resultat auch rNoRec zurückgegeben werden. In diesem Fall ist in der Datei (int1) die maximale Anzahl von 1000 Datensätzen erreicht.

Beispiele

```
RecBufClear(tblCstCustomer);... // Datensatzpuffer fülltErg # RecInsert(tblCstCustomer, 0);// M
```

Mögliche Laufzeitfehler:

ErrNoFile Datei nicht vorhanden

RecLink(int1, int2, int3,
int4[,int5[,int6]]) : int




Verknüpften Datensatz lesen

int1 Dateinummer / Deskriptor des
 Datensatzpuffers (Zieldatei)
int2 Dateinummer / Deskriptor des
 Datensatzpuffers (Ausgangsdatei)
int3 Verknüpfungsnummer /
 Selektions-Deskriptor
int4 Optionen (siehe Text)
int5 Filter-Deskriptor (optional)
int6 Verknüfungsposition (optional)

Leseresultat

rOk Lesen erfolgreich
rLocked Verknüpfter
 Datensatz gesperrt
rNoRec Kein (weiterer)
 verknüpfter
 Datensatz
 vorhanden

Resultat int rNoRights Benutzerrechte
 nicht ausreichend 
rFailed Zugriff auf
 temporären Baum
 nicht möglich
 (Clients < 5.8.11
 erhalten rNoRec).

ErrDbComm Verbindungsabbruch
 zu einer
 verbundenen
 Datenbank
 aufgetreten.

Siehe Verwandte Befehle, RecRead(), SelOpen(),
Datensatzzugriff, Beispiel

Mit RecLink() sind verschiedene Zugriffsmöglichkeiten auf verknüpfte Datensätze realisiert.

In (int1) wird die Zieldatei der Verknüpfung angegeben. Diese Angabe dient zur Kontrolle der korrekten Verknüpfung und zur Selbstdokumentation. Ist als Parameter (int3) eine Variable angegeben (unterschiedliche Zieldateien), so kann in (int1) auch 0 übergeben werden (nicht bei Angabe einer Selektion).

In (int2) wird die Nummer der Ausgangsdatei übergeben, in (int3) entweder die Verknüpfungsnummer oder der Deskriptor einer offenen Selektion (siehe SelOpen()). Werden in (int1) und/oder (int2) Deskriptoren von Datensatzpuffern angegeben, erfolgt das Lesen und Schreiben in diese Datensatzpuffer.

Sofern keine Optionen in (int4) angegeben werden, ist das Verhalten folgendermaßen:

Aus den im Hauptspeicher befindlichen Verknüpfungsfeldern (inkl. der Felder für Zugriffspositionierung) der in (int3) angegebenen Verknüpfung wird ein Schlüsselwert gebildet und auf den Satz in der Zielfeile zugegriffen.

Anstatt einer Verknüpfung kann in (int3) auch der Deskriptor einer Selektion angegeben werden. Dabei muss sich eine der verknüpften Ergebnismengen auf die Zielfeile beziehen.

Die Optionen in (int4) teilen sich in drei unterschiedliche Gruppen:

Zugriffsoptionen

_RecFirst

Der erste verknüpfte Satz wird geladen. Dabei wird eine eventuell definierte Zugriffspositionierung ignoriert.

_RecLast

Der letzte verknüpfte Satz wird geladen.

_RecNext

Der nächste verknüpfte Satz wird geladen. Sofern kein weiterer verknüpfter Satz vorhanden ist, wird als Resultat _rNoRec zurückgeliefert.

_RecPrev

Der vorherige verknüpfte Satz wird geladen. Sofern kein weiterer verknüpfter Satz vorhanden ist, wird als Resultat _rNoRec zurückgeliefert.

_RecKeyReverse

Die Schlüsselreihenfolge in der mit _RecPrev, _RecNext und _RecPos zugegriffen wird, kann mit dieser Konstanten umgekehrt werden.

_RecPos

Der Zugriff findet über die Verknüfungsposition statt (siehe RecLinkInfo()), die in (int6) übergeben werden muss. Da die Position exakt verarbeitet wird, liegen gültige Werte im Bereich von 1 bis zur Anzahl verknüpfter Sätze, andernfalls wird _rNoRec zurückgeliefert.

Sperroptionen

Wird keine Sperroption angegeben, so bleibt der Sperrstatus des gelesenen Satzes unverändert.

_RecUnlock

Der gelesene Satz wird entsperrt.

RecLock

Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn rOk zurückgeliefert wird. Ist der Satz durch einen anderen Benutzer gesperrt, so ist das Resultat rLocked.

RecSharedLock

Der gelesene Datensatz wird gemeinsam mit anderen Benutzern gesperrt. Dies ist nur dann der Fall, wenn rOk zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, so ist das Resultat rLocked. Im Gegensatz zur Option RecLock können von anderen Benutzern weitere Sperren mit dieser Option eingerichtet werden. Ein mit RecSharedLock gesperrter Datensatz kann von einem anderen Benutzer nicht mit der Option RecLock zum Schreiben gesperrt werden, bis die letzte Sperre aufgehoben wurde. Der Benutzer, der die gemeinsame Sperre eingerichtet hat, kann den Datensatz nur dann mit RecLock sperren, wenn in der Zwischenzeit kein anderer Benutzer eine gemeinsame Sperre eingerichtet hat. Mit dieser Option können mehrere Benutzer einen Datensatz vor Veränderung schützen. Der Datensatz kann mit dieser Sperre nicht zurückgeschrieben werden.

RecForceLock

Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn rOk zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, kann dieser Benutzer den Satz nicht mehr zurückspeichern. Daher sollte RecForceLock nur in den Fällen erfolgen, in denen ein Satz unbedingt und ohne Rücksicht auf andere Benutzer gesperrt werden muss. Die Sperre schlägt fehl, wenn der Datensatz durch andere Benutzer gemeinsam (RecSharedLock) gesperrt ist oder der Datensatz in diesem Moment gerade geschrieben oder gelöscht wird. Wird rLocked zurückgegeben, kann der Befehl nach kurzer Zeit erneut versucht werden.



Fahrlässiger Umgang mit dieser Option kann zur Inkonsistenz des Datenbestandes führen.

RecCheckLock

Der Sperrstatus des Satzes wird überprüft. Das Resultat ist rLocked anstelle von rOk, wenn ein anderer Benutzer den Satz gesperrt hat. Der Sperrstatus wird nicht verändert.

Ladeoptionen

RecTest

Hierbei erfolgt der Zugriff nur auf den Schlüssel der Zieldatei. Es wird nicht auf den Datensatz zugegriffen, daher ist die Angabe einer Sperroption ohne Wirkung.

RecTest kann nicht mit den Optionen RecPrev und RecNext kombiniert werden.

RecNoLoad

Kontakt


Der gelesene Datensatz wird nicht in die Feldpuffer übertragen.

In (int5) kann der Deskriptor eines Filters übergeben werden (siehe RecFilterCreate()). Dabei muss der Filter auf die Zielfeld und den durch die Verknüpfung benutzten Schlüssel gelegt sein.

In (int6) wird die Verknüfungsposition bei Verwendung von _RecPos angegeben.

Mögliche Laufzeitfehler:

<u>_ErrNoFile</u>	Die Datei (int1) bzw. (int2) ist nicht vorhanden.
<u>_ErrNoLink</u>	Die Verknüpfung (int3) ist nicht vorhanden.
<u>_ErrHdlInvalid</u>	Der Selektions- (int3) oder Filter-Deskriptor (int5) ungültig.
<u>_ErrNoArgument</u>	Das Argument (int5) oder (int6) wird benötigt, ist aber nicht angegeben.
<u>_ErrLinkInvalid</u>	Die Verknüpfung (int3) passt nicht zu Zielfeld (int1) oder Filter (int5).
<u>_ErrFileInvalid</u>	Die Zielfeld (int1) passt nicht zur Verknüpfung der Selektion (int3).

RecLinkInfo(int1, int2, int3, int4[,int5]) : int 

Verknüpfungsdatensatzinformationen ermitteln

- int1 Dateinummer oder Deskriptor eines Datensatzpuffers (Zieldatei)
- int2 Dateinummer oder Deskriptor eines Datensatzpuffers (Ausgangsdatei)
- int3 Verknüpfungsnummer / Selektions-Deskriptor
Informationstyp
 - _RecCount Datensatzanzahl ermitteln
 - _RecCountNext Anzahl der Folgedatensätze ab aktueller Position ermitteln
 - _RecCountPos Anzahl der Folgedatensätze ab bestimmter Position ermitteln
 - _RecGetPos Schlüssel-/Verknüfungsposition ermitteln
- int5 Filter-Deskriptor (optional)

Resultat int Verknüpfungsinformation

Siehe Verwandte Befehle, RecBufCreate()

Diese Funktion ermittelt Informationen zu verknüpften Datensätzen. In (int1) wird die Zieldatei der Verknüpfung oder ein Deskriptor auf einen Datensatzpuffer der Zeildatei angegeben. Diese Angabe dient zur Kontrolle der korrekten Verknüpfung und zur Selbstdokumentation. Ist als Parameter (int3) eine Variable angegeben (unterschiedliche Zieldateien), kann in (int1) auch 0 übergeben werden (nicht bei Angabe einer Selektion).

In (int2) wird die Nummer oder der Deskriptor auf einen Datensatzpuffer der Ausgangsdatei übergeben, in (int3) entweder die Verknüpfungsnummer oder der Deskriptor einer offenen Selektion (siehe SelOpen()).

Folgende Informationen können ermittelt werden (int4):

_RecCount

Es wird die Anzahl aller verknüpften Datensätze zurückgeliefert. Eine eventuelle Zugriffspositionierung wird dabei ignoriert.

_RecCountNext

Die Anzahl verknüpfter Sätze nach dem aktuellen Satz in der Zieldatei wird ermittelt. Das Resultat ist -1, wenn dieser Satz nicht zur verknüpften Menge gehört.

_RecCountPos

Es wird die Anzahl aller verknüpften Datensätze ab der definierten Zugriffsposition zurückgeliefert. Der per Zugriffspositionierung verknüpfte Satz wird mitgezählt.

_RecGetPos

Kontakt

Die Position des aktuellen Satzes der Zielfeile wird ermittelt. Der erste verknüpfte Satz hat dabei die Nummer 1. Das Resultat ist -1, wenn dieser Satz nicht zur verknüpften Menge gehört. Im Gegensatz zu RecInfo() ist das Resultat exakt, dafür kann die Funktion bei einer sehr großen Anzahl verknüpfter Sätze eine entsprechend lange Verarbeitungszeit brauchen.


RecGetPosReverse

Die umgekehrte Position des aktuellen Satzes der Zielfeile wird ermittelt. Der letzte verknüpfte Satz hat dabei die Nummer 1. Das Resultat ist -1, wenn dieser Satz nicht zur verknüpften Menge gehört. Im Gegensatz zu RecInfo() ist das Resultat exakt, dafür kann die Funktion bei einer sehr großen Anzahl verknüpfter Sätze eine entsprechend lange Verarbeitungszeit brauchen.

In (int5) kann optional ein Filter-Deskriptor angegeben werden. Dabei muss der Filter auf den in der verknüpften Feile benutzten Schlüssel gelegt sein. Ein Filter kann nicht bei Angabe einer Selektion benutzt werden.

Mögliche Laufzeitfehler:

<u>ErrNoFile</u>	Die Feile (int1) bzw. (int2) ist nicht vorhanden.
<u>ErrNoLink</u>	Die Verknüpfung (int3) ist nicht vorhanden.
<u>ErrHdlInvalid</u>	Der Selektions- (int3) oder Filter-Deskriptor (int5) ungültig.
<u>ErrLinkInvalid</u>	Die Verknüpfung (int3) passt nicht zu Zielfeile (int1) oder Filter (int5).
<u>ErrFileInvalid</u>	Die Zielfeile (int1) passt nicht zur Verknüpfung der Selektion (int3).


RecRead(int1, int2, int3[,  bigint4[, int5]]) : int

Datensatz lesen

int1 Dateinummer oder Deskriptor eines
Datensatzpuffers
int2 Schlüsselnummer / Selektions-Deskriptor
int3 Optionen (siehe Text)
bigint4 Datensatzposition / Filter-Deskriptor /
Datensatz-ID (optional)
int5 Anzahl der Sätze bei RecPrev und
RecNext (optional)

Leseresultat

rOk Lesen erfolgreich
rLocked Datensatz gesperrt
rMultiKey Schlüssel nicht
eindeutig
rNoKey Schlüssel nicht
vorhanden, nächsten
Datensatz geladen
rLastRec Schlüssel nicht
vorhanden, letzten
Datensatz geladen
rNoRec Kein (weiterer)
Datensatz
vorhanden

Resultat int rNoLock Datensatz ist nicht 
gesperrt
rNoRights Benutzerrechte
nicht ausreichend
rDeadlock Verklemmung
aufgetreten
rFailed Zugriff auf
temporären Baum
nicht möglich
(Clients < 5.8.11
erhalten rNoRec).
ErrDbComm Verbindungsabbruch
zu einer
verbundenen
Datenbank
aufgetreten.

Siehe Verwandte Befehle, RecBufCreate(),
RecLink(), SelOpen(), Datensatzpuffer,
Datensatzzugriff

Mit RecRead() sind verschiedene Zugriffsmöglichkeiten auf Datensätze realisiert.

Sofern keine Optionen in (int3) angegeben werden, ist das Verhalten folgendermaßen:

Dem Befehl werden die Datei (int1) und der Schlüssel (int2) übergeben. In dem Schlüssel sind die Felder des Datensatzes definiert, aus denen die Schlüsselwerte gebildet werden. Beim Zugriff auf einen Datensatz wird aus dem Inhalt dieser Felder der Schlüsselwert erstellt und in der Schlüsseltabelle gesucht. Anhand des erhaltenen Schlüssels wird dann ein Datensatz in den Standard-Datensatzpuffer geladen. Ist in (int1) ein Datensatzpuffer angegeben (siehe RecBufCreate()), wird dieser beim Laden verwendet.

Anstatt eines Schlüssels kann in (int2) auch der Deskriptor einer Selektion angegeben werden (siehe SelOpen()). Dabei muss sich die Hauptergebnismenge der Selektion auf die in (int1) angegebene Datei beziehen und die Selektionsmenge sperrend gelesen worden sein. Für einen Zugriff über die Datensatz-ID wird in (int2) 0, in (int3) RecID und in (bigint4) die Datensatz-ID übergeben.

Mehrere Optionen können miteinander kombiniert werden. Die Kombination erfolgt durch eine binäre ODER-Verknüpfung.

Beispiel:

```
RecRead(Adr.D.Adressen, Adr.S.iNummer, _RecFirst | _RecLock);
```

Die Optionen in (int3) teilen sich in drei unterschiedliche Gruppen:

Lesen von Datensätzen

- **0**

Ausgehend vom Inhalt der Feldpuffer wird genau der angegebene Datensatz gelesen. Der Rückgabewert des Befehls ist dann abhängig, ob über einen eindeutigen oder mehrdeutigen Schlüssel gelesen wurde rOk oder rMultiKey. Eine Satzsperrung beeinflusst den Rückgabewert nicht. Konnte aus dem Inhalt der Schlüsselfelder kein vorhandener Schlüsselwert gebildet werden, wird der darauf folgende Datensatz gelesen. In diesem Fall gibt der Befehl den Wert rNoKey zurück. Gibt es keinen nachfolgenden Datensatz, wird der letzte Datensatz gelesen. Zurückgegeben wird dann der Wert rLastRec. Konnte kein Datensatz gelesen werden (d. h. die Datei ist leer), wird der Wert rNoRec zurückgegeben.

- **RecFirst**

Der Satz mit dem kleinsten Schlüsselwert wird geladen.

- **RecLast**

Der Satz mit dem größten Schlüsselwert wird geladen.

- **RecNext**

Ausgehend vom Inhalt der Feldpuffer wird der Satz mit dem nächst größeren Schlüsselwert geladen. Sofern kein weiterer Satz vorhanden ist, wird als Resultat rNoRec zurückgeliefert.

- **RecPrev**

Ausgehend vom Inhalt der Feldpuffer wird der Satz mit dem nächst kleineren Schlüsselwert geladen. Sofern kein weiterer Satz vorhanden ist, wird als Resultat rNoRec zurückgeliefert.

- **RecKeyReverse**

Die Schlüsselreihenfolge in der mit RecPrev, RecNext und RecPos zugegriffen wird, kann mit dieser Konstanten umgekehrt werden.

- **RecID**

Anstatt über einen Schlüssel wird der Datensatz über die Datensatz-ID gelesen. In (int2) darf kein Schlüssel oder Selektionsdeskriptor übergeben werden. Die Datensatz-ID wird entweder durch das aufgerufene Ereignis (wie zum Beispiel bei EvtLstSelect) übergeben oder wurde zuvor mit der Anweisung RecInfo(..., RecID) ermittelt. Im Parameter (bigint4) muss die zu lesende Datensatz-ID übergeben werden.

- **RecPos**

Anstatt über den aktuellen Schlüsselwert wird über die Schlüsselposition zugegriffen (siehe RecInfo()), die in (bigint4) übergeben werden muss. Diese Position wird aus nur ungefähr berechnet, da eine genaue Positionierung zu lange dauern würde.

Optional kann bei Verwendung von RecPrev oder RecNext in (int5) die Anzahl vorheriger bzw. nächsten Sätze angegeben werden, die gelesen werden sollen. Wird in (int5) beispielsweise 5 angegeben, wird in der Schlüsselreihe um 5 Sätze nach vorn bzw. zurück positioniert. Sind weniger Sätze als die angegebene Anzahl vorhanden, wird der erste bzw. der letzte Satz gelesen und das Resultat ist rNoKey.

Beispiele:

```
// Lesen des Datensatzes mit der Nummer 1000KND.iID # 1000;tErg # RecRead(KND.D.Kunden, KND.S.ID,
```

Sperroptionen

Wird keine Sperroption angegeben, so bleibt der Sperrstatus des gelesenen Satzes unverändert.

- **RecUnlock**

Der gelesene Satz wird entsperrt.

- **RecLock**

Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn rOk zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, ist das Resultat rLocked und der Datensatz wurde nicht gesperrt. Tritt beim Sperren eine Verklemmung auf, wird rDeadlock zurückgegeben und der Datensatz wird nicht gesperrt.

- **RecSingleLock**

Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn rOk zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch den eigenen

oder einen anderen Benutzer gesperrt, ist das Resultat rLocked. Das Sperren eines Datensatzes erfolgt über die Benutzer-ID. Mit dem Parameter RecSingleLock wird der gleiche Datensatz für den gleichen Benutzer nur einmal gesperrt. Beim Versuch, denselben Satz ein zweites Mal zu sperren, wird das Resultat rLocked zurückgeliefert. Diese Option muss in einer Applikation verwendet werden, in der zum Beispiel mehrere MDI-Fenster den gleichen Datensatz sperren können. Tritt beim Sperren eine Verklemmung auf, wird rDeadlock zurückgegeben und der Datensatz wird nicht gesperrt.

- **RecSharedLock**

Der gelesene Datensatz wird gemeinsam mit anderen Benutzern gesperrt. Dies ist nur dann der Fall, wenn rOk zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, ist das Resultat rLocked. Im Gegensatz zur Option RecLock können von anderen Benutzern weitere Sperren mit dieser Option eingerichtet werden. Ein mit RecSharedLock gesperrter Datensatz kann von einem anderen Benutzer nicht mit der Option RecLock zum Schreiben gesperrt werden, bis die letzte Sperre aufgehoben wurde. Der Benutzer, der die gemeinsame Sperre eingerichtet hat, kann den Datensatz nur dann mit RecLock sperren, wenn in der Zwischenzeit kein anderer Benutzer eine gemeinsame Sperre eingerichtet hat. Mit dieser Option können mehrere Benutzer einen Datensatz vor Veränderung schützen. Der Datensatz kann mit dieser Sperre nicht zurückgeschrieben werden. Tritt beim Sperren eine Verklemmung auf, wird rDeadlock zurückgegeben und der Datensatz wird nicht gesperrt.

- **RecForceLock**

Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn rOk zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, kann dieser Benutzer den Satz nicht mehr zurückspeichern. Daher sollte RecForceLock nur in den Fällen erfolgen, in denen ein Satz unbedingt und ohne Rücksicht auf andere Benutzer gesperrt werden muss. Die Sperre schlägt fehl, wenn der Datensatz durch andere Benutzer gemeinsam (RecSharedLock) gesperrt ist oder der Datensatz in diesem Moment gerade geschrieben oder gelöscht wird. Wird rLocked zurückgegeben, kann der Befehl nach kurzer Zeit erneut versucht werden. Tritt beim Sperren eine Verklemmung auf, wird rDeadlock zurückgegeben und der Datensatz wird nicht gesperrt.



Fahrlässiger Umgang mit dieser Option kann zur Inkonsistenz des Datenbestandes führen.

- **RecCheckLock**

Der Sperrstatus des Satzes wird überprüft. Das Resultat ist rLocked, wenn ein anderer Benutzer den Satz gesperrt hat, sonst rOk. Der Sperrstatus wird nicht verändert.

Die Sperre eines Datensatzes wird an seiner Datensatz-ID festgehalten. D. h. es können alle Felder eines Datensatzes geändert werden, auch die des eindeutigen Schlüssels. Eine Änderung der Datensatz-ID zum Beispiel durch RecInfo(..., RecIdSet, ...), darf nicht erfolgen. Wurde Datensatz-ID geändert, gibt die Anweisung

RecReplace() den Fehler _rNoLock zurück.



Wurde ein Datensatz gesperrt, bleibt er solange gesperrt, bis zum Beispiel die Anweisung RecRead(..., _RecUnlock) den Datensatz wieder entsperrt oder sich der Benutzer aus der Datenbank abmeldet. Werden Datensätze durch das Abmelden eines Benutzers entsperrt, kann sich das Abmelden entsprechend verzögern.

Beispiele:

```
// Datensatz mit der Nummer 1000 Lesen und sperrenfiCstID # 1000;tErg # RecRead(tblCustomer, keyC
```

Ladeoptionen

- **RecTest**

Hierbei erfolgt der Zugriff nur auf den Schlüssel. Es wird nicht auf den Datensatz zugegriffen, daher ist die Angabe einer Sperroption ohne Wirkung.

- **RecNoLoad**

Der gelesene Datensatz wird nicht in die Feldpuffer übertragen.

Ist der angegebene Schlüsselwert vorhanden wird in beiden Fällen abhängig davon, ob über einen ein- oder mehrdeutigen Schlüssel zugegriffen wurde _rOk oder _rMultiKey zurückgegeben. Ist der Schlüsselwert nicht vorhanden, wird _rNoKey oder _rLastRec zurückgegeben. Je nachdem, ob es einen nachfolgenden Schlüsselwert gab oder nicht.

Beim Lesen eines Datensatzes mit diesen Ladeoptionen wird die Datensatz-ID nicht gesetzt. Vor einem Zurückschreiben des Satzes mit dem Befehl RecReplace() muss aber eine Datensatz-ID entweder durch Lesen des Datensatzes oder durch Setzen der ID mit dem Befehl RecInfo() vorhanden sein.

Beispiele

```
// Den Datensatz mit der Nummer 1000 sperren, ohne die anderen Feldpuffer zu überschreibenfiCstID
```

Das Argument (bigint4) ist optional und hat drei verschiedene Bedeutungen:

- **Schlüsselposition**

Bei Benutzung der Option _RecPos wird in (bigint4) die gewünschte Schlüsselposition übergeben.

- **Datensatz-ID**

Ist (int2) gleich 0, wird in (bigint4) die ID des gewünschten Datensatzes übergeben. Die Datensatz-ID entspricht einem eindeutigen Schlüssel, der für interne Zwecke verwendet wird. Sie wird ebenfalls bei dem Ereignis EvtLstSelect übergeben.

Die Optionen _RecTest und _RecPos sowie die Angabe eines Schlüssels oder einer Selektion sind dabei nicht möglich, da kein Schlüsselzugriff erfolgt.

Dadurch gibt es auch die Resultate _rNoKey und _rMultiKey nicht.

Ist der Datensatz mit der angegebenen ID nicht vorhanden, wird _rNoRec zurückgeliefert und kein Satz geladen.

Wird eine Datensatz-ID angegeben, kann nicht mit den Optionen _RecFirst, _RecLast, _RecNext oder _RecPrev zugegriffen werden.

- **Filter-Deskriptor**

Sofern weder _RecPos noch die Datensatz-ID verwendet wird, kann in (bigint4) der Deskriptor eines Filters angegeben werden (siehe RecFilterCreate()).


Resultate

Das Resultat von RecRead() kann durch die aufsteigenden Resultatwerte in den meisten Fällen durch einen Größer-/Kleiner-Vergleich ausgewertet werden:

```
// Feststellen ob ein Satz geladen wurde if (RecRead(...) < _rNoRec){ ...}
```

Mögliche Laufzeitfehler:

<u>_ErrNoFile</u>	Datei nicht vorhanden
<u>_ErrNoKey</u>	Schlüssel nicht vorhanden
<u>_ErrHdlInvalid</u>	Filter- oder Selektions-Deskriptor ungültig
<u>_ErrNoArgument</u>	Argument (bigint4) wird benötigt, fehlt aber

RecReplace(int1, int2) : int 

Datensatz ersetzen

int1 Dateinummer oder Deskriptor eines
 Datensatzpuffers
 Optionen
 0 Datensatzsperre
 beibehalten


int2 __RecUnlock Datensatz entsperren
 __RecEarlyCommit Änderung trotz
 Transaktion
 durchführen

Einfügeresultat

__rOk Ersetzen erfolgreich

__rExists Datensatz mit
 identischen
 eindeutigen
 Schlüsselwerten
 existiert bereits

__rNoRec Datensatz mit der
 aktuellen
 Datensatz-ID nicht
 vorhanden

__rNoLock Datensatz mit der
 aktuellen
 Datensatz-ID nicht
 gesperrt 

Resultat __rNoRights Benutzerrechte
 nicht ausreichend

__rDeadlock Verklemmung
 aufgetreten

__rFailed Zugriff auf
 temporären Baum
 nicht möglich
 (Clients < 5.8.11
 erhalten __rNoRec).

__ErrDbComm Verbindungsabbruch
 zu einer
 verbundenen
 Datenbank
 aufgetreten.

Verwandte Befehle, RecDelete(),

Siehe RecInsert(), RecBufCreate(),
Datensatzpuffer

Diese Funktion speichert die momentan im Speicher befindlichen Feldpuffer einer Datei (bestehend aus den aktuellen Feldinhalten) in die Datenbank zurück. Der entsprechende Datensatz muss vorher geladen und gesperrt worden sein. Ein nicht gesperrter Satz kann nicht zurückgespeichert werden. Alle Felder des gelesenen Satzes können vor dem Rückspeichern verändert werden (inkl. aller Schlüsselfelder).

Kontakt

Dabei ist zu beachten, dass eindeutige Schlüsselwerte nicht bereits in der Datei enthalten sein dürfen. Wird in (int1) die Dateinummer angegeben, wird der Inhalt der Feldpuffer der Datei gespeichert. Es kann auch der Deskriptor eines mit RecBufCreate() angelegten Datensatzpuffers angegeben werden. Dann wird der Inhalt des übergebenen Datensatzpuffers in die dazugehörige Datei übertragen.

Durch die Option (int2) kann angegeben werden, ob der Datensatz nach dem Rückspeichern weiter gesperrt bleiben soll (0), oder die Sperre aufgehoben werden soll (_RecUnlock).

Bei der Angabe von _RecEarlyCommit können innerhalb einer Transaktion (siehe DtaBegin()) Änderungen am Datenbestand vorgenommen werden, ohne dass es zu Wartezuständen bei anderen Transaktionen kommt.

Das Ergebnis _rExists bedeutet, dass bereits ein Datensatz mit einem gleichen eindeutigen Schlüssel vorhanden ist und somit der Datensatz nicht zurückgespeichert werden konnte. Er bleibt daher gesperrt.

Sollte der Datensatz nicht gesperrt sein, so ist als Ergebnis _rNoLock möglich.

Mögliche Laufzeitfehler:

_ErrNoFile Datei nicht vorhanden

Konstanten für Datensatzbefehle

Konstanten für Datensatzbefehle

Siehe Datensatzbefehle

<u>RecCheckLock</u>	Datensatzsperre überprüfen
<u>RecCount</u>	Datensatzanzahl ermitteln
<u>RecCountNext</u>	Anzahl der Folgedatensätze ab aktueller Position ermitteln
<u>RecCountPos</u>	Anzahl der Folgedatensätze ab bestimmter Position ermitteln
<u>RecEarlyCommit</u>	Änderung trotz Transaktion durchführen
<u>RecFirst</u>	Ersten Datensatz ermitteln
<u>RecForceLock</u>	Datensatz zwingend sperren
<u>RecGetPos</u>	Schlüssel-/Verknüpfungsposition ermitteln
<u>RecID</u>	<u>Datensatz-ID</u> ermitteln
<u>RecLen</u>	Datensatzgröße ermitteln
<u>RecLenPacked</u>	Datensatzgröße in der Datenbank ermitteln
<u>RecLast</u>	Letzten Datensatz ermitteln
<u>RecLock</u>	Datensatz sperren
<u>RecLockedBy</u>	Sperrenden Benutzer ermitteln
<u>RecNext</u>	Nächsten Datensatz ermitteln
<u>RecNoLoad</u>	Datensatz ohne Feldpufferübertragung lesen
<u>RecPos</u>	Datensatz an bestimmter Schlüssel-/Verknüpfungsposition lesen
<u>RecPrev</u>	Vorherigen Datensatz ermitteln
<u>RecSetID</u>	<u>Datensatz-ID</u> setzen
<u>RecSharedLock</u>	Datensatz gemeinsam sperren
<u>RecSingleLock</u>	Datensatz einfach sperren
<u>RecTest</u>	Datensatzexistenz überprüfen
<u>RecUnlock</u>	Datensatz entsperren
<u>RecUpdateCounter</u>	Update-Zähler ermitteln

_RecCheckLock
Sperrstatus ermitteln

Wert 24 /
0x00000018

Verwandte

Siehe Befehle,
RecRead(),
RecLink()

Option bei RecRead() und RecLink() durch die eine Datensatzsperre ermittelt werden kann. Das Resultat ist _rLocked, wenn ein anderer Benutzer den Satz gesperrt hat, sonst _rOk. Der Sperrstatus wird nicht verändert.

_RecCount
Datensatzanzahl ermitteln
Wert 0

Verwandte
Siehe Befehle,
RecInfo(),
RecLinkInfo()

Option bei RecInfo() und RecLinkInfo() durch die die Anzahl der (verknüpften) Datensätze einer Datei ermittelt werden kann.

RecCountNext
Anzahl der Folgedatensätze ab aktueller Position ermitteln
Wert 6

Verwandte

Siehe Befehle,

RecLinkInfo()

Option bei RecLinkInfo() durch die die Anzahl der verknüpften Folgedatensätze ab der aktuellen Zugriffsposition ermittelt werden kann.

RecCountPos

Anzahl der Folgedatensätze ab bestimmter Position ermitteln

Wert 5

Verwandte

Siehe Befehle,

RecLinkInfo()

Option bei RecLinkInfo() durch die die Anzahl der verknüpften Folgedatensätze ab einer bestimmten Zugriffsposition ermittelt werden kann.

_RecEarlyCommit
Änderung trotz Transaktion durchführen

Wert 1.073.741.824
/ 0x40000000

Verwandte

Befehle,

Siehe DtaBegin(),
RecDelete(),
RecInsert(),
RecReplace()

Option bei RecDelete(), RecInsert() und RecReplace().

Normalerweise werden alle Änderungen innerhalb einer Transaktion erst am Ende der Transaktion wirksam. Dadurch bleiben aber auch die beteiligten Segmente in der Datenbank bis zum Ende gesperrt, wodurch parallel laufende Transaktionen unter Umständen behindert werden können.

Bei Verwendung von _RecEarlyCommit wird die einzelne Änderung sofort wirksam, die entsprechenden Segmentsperren werden wieder aufgehoben. Allerdings werden dann diese Änderungen bei einem Transaktionsabbruch (siehe DtaRollback()) nicht wieder rückgängig gemacht.

Sofern bei einer Operation mit _RecEarlyCommit ein Datensegment berührt wird, welches in der Transaktion bereits verändert wurde, ist die Option _RecEarlyCommit nicht wirksam (daher sollte die Option nicht bei Änderungen in denselben Dateien benutzt werden).

Beispiel:

```
DtaBegin();RecInsert(tblOrdOrders, ...)...RecInsert(tblPrtProtocol, _RecEarlyCommit);...DtaCommit
```

In diesem Beispiel wird das Einfügen in die Protokolldatei sofort wirksam, dadurch entfallen mögliche Wartezustände oder Verklemmungen bei paralleler Verwendung der Protokolldatei. Bei einem DtaRollback() bleibt die Änderung in der Protokolldatei trotzdem erhalten.

RecFirst
Erster Datensatz
Wert 1 /
0x00000001

Verwandte
Befehle,

Siehe RecRead(),
RecLink(),
RecDelete()


Option bei RecRead(), RecLink() und RecDelete() durch die der erste (verknüpfte) Datensatz einer Datei gelesen/gelöscht werden kann.

_RecGetPos
Schlüssel-/Verknüpfungsposition ermitteln
Wert 4

Verwandte Befehle,
Siehe RecInfo(),
RecLinkInfo(),
_RecGetPosReverse

Option bei RecInfo() und RecLinkInfo() durch die die Schlüssel-/Verknüpfungsposition des aktuellen Datensatzes ermittelt werden kann.

Ausgehend von einer Datei mit 100 Datensätze, die alle hintereinander durchnummeriert sind und der Datensatz mit der Nummer 4 geladen wurde, wird als Ergebnis von `RecInfo(_RecGetPos)` 4 zurückgegeben.

 Bei RecInfo() wird nur die ungefähre Position ermittelt, da eine exakte Positionsermittlung zu lange dauern würde (bedingt durch variable Schlüssellängen und dynamisches Speichermanagement in der Datenbank).

_RecGetPosReverse
Schlüssel-/Verknüpfungsposition ermitteln
Wert 8

Verwandte

Befehle,

Siehe RecInfo(),
RecLinkInfo(),
_RecGetPos

Option bei RecInfo() und RecLinkInfo() durch die die Schlüssel-/Verknüpfungsposition des aktuellen Datensatzes ermittelt werden kann. Die Position wird über die umgekehrte Schlüsselsortierung ermittelt.

Ausgehend von einer Datei mit 100 Datensätze, die alle hintereinander durchnummeriert sind und der Datensatz mit der Nummer 4 geladen wurde, wird als Ergebnis von `RecInfo(_RecGetPos)` 97 zurückgegeben.



Bei RecInfo() wird nur die ungefähre Position ermittelt, da eine exakte Positionsermittlung zu lange dauern würde (bedingt durch variable Schlüssellängen und dynamisches Speichermanagement in der Datenbank).

_RecGetPrime
Prime-Counter ermitteln
Wert 9

Verwandte

Siehe Befehle,
RecInfo()

Option bei RecInfo() durch die der Prime-Counter einer Datei ermittelt werden kann.
Der Prime-Counter ist die höchste Datensatz-ID, die in dieser Tabelle vergeben wurde.

_RecID
Datensatz-ID ermitteln
Wert 1

Verwandte
Befehle,
Siehe RecRead(),
RecInfo(),
c16_recread(),
c16_recinfo()

Option bei RecRead(), RecLink() und c16_recread() durch die der Datensatz mit einer bestimmten Datensatz-ID gelesen werden kann (bei RecRead()) bzw. die Datensatz-ID des aktuellen Datensatzes ermittelt werden kann (bei RecInfo()).



Bei der Datensatz-ID handelt es sich um eine interne ID, die sich durch bestimmte Aktionen (z. B. Export und Import der Datensätze) ändern kann. Sie sollte daher nicht als dauerhafte Referenz auf einen Datensatz verwendet werden.

_RecKeyReverse
Umgekehrte Sortierung des Schlüssels verwendet

Wert 2.097.152 /
0x00200000

Verwandte

Siehe Befehle,
RecRead(),
RecLink()

Option bei dem Befehl RecRead() und RecLink(). Wird diese Option angegeben, wird die Schlüsselreihenfolge umgedreht. Dies wirkt sich auf den Zugriff der Datensätze aus.

Im Gegensatz zum Schlüsselfeld-Attribut "absteigende Sortierung" (siehe Schlüssel bearbeiten) wird nicht die Reihenfolge in einem Schlüsselfeld, sondern die Reihenfolge in allen Schlüsselfeldern umgekehrt.

Reihenfolge ohne

_RecKeyReverse

CH	3000
CH	4589
CH	4658
CH	6000
D	05895
D	10258
D	25598
D	63150
D	70597

Reihenfolge mit

_RecKeyReverse

D	70597
D	63150
D	25598
D	10258
D	05895
CH	6000
CH	4658
CH	4589
CH	3000

RecLast
Letzter Datensatz
Wert 2 /
0x00000002

Verwandte

Befehle,

Siehe RecRead(),

RecLink(),

RecDelete()

Option bei RecRead(), RecLink() und RecDelete() durch die der letzte (verknüpfte) Datensatz einer Datei gelesen/gelöscht werden kann.

_RecLen
Datensatzgröße ermitteln
Wert 2

Verwandte
Siehe Befehle,
RecInfo(),
_RecLenPacked

Option bei RecInfo() durch die die Größe eines Datensatzes ermittelt werden kann.

_RecLenPacked
Datensatzgröße in der Datenbank ermitteln
Wert 10

Verwandte
Siehe Befehle,
RecInfo(),
_RecLen

Option bei RecInfo() durch die die gepackte Größe eines Datensatzes, d. h. die Größe des Datensatzes in der Datenbank, ermittelt werden kann. Änderungen am Feldpuffer werden nicht berücksichtigt. In der gepackten Länge sind alphanumerische Felder nur mit ihrer aktuellen Länge berücksichtigt. Pro Feld wird ein Bit gesetzt, welches entscheidet, ob das Feld einen Wert hat. Leere Felder nehmen zusätzlich zu diesem Bit keinen Platz ein. Bei logic-Feldern entscheidet das Bit über den Wert und nicht über das Vorhandensein eines Wertes.

 Die gepackte Größe wird nur nach dem Lesen (z. B. RecRead()) und Ändern (z. B. RecInsert()) aktualisiert.

_RecLockedBy
Sperrenden Benutzer ermitteln
Wert 3

Verwandte

Siehe Befehle,
RecInfo()

Option bei RecInfo() durch die die Benutzer-ID des sperrenden Benutzers ermittelt werden kann.

RecModified
Zeitpunkt der Änderung ermitteln
Wert 14

Verwandte
Befehle,
Siehe RecRead(),
RecInfo(),
c16_recread(),
c16_recinfo()

Option bei RecInfo() womit der Änderungszeitpunkt des aktuellen Datensatzes ermittelt werden kann.

Damit Datensätze mit Änderungszeitpunkt gespeichert werden, muss die entsprechende Option im Datenstruktureditor gesetzt sein. Sofern die Option gesetzt ist, werden Datensätze beim Einfügen bzw. Ändern eines Datensatzes durch RecInsert() bzw. RecReplace() mit einem Zeitstempel versehen. Der Zeitstempel entspricht der aktuellen Zeit des Datenbank-Servers (UTC).

Damit der Zeitstempel ausgelesen werden kann, muss nach der Änderung erst ein RecRead() erfolgen. Der Zeitstempel kann bei Bedarf über CnvAB() in eine Zeichenkette konvertiert werden.



Der Zeitstempel wird im Falle eines RecReplace() auch dann gesetzt, wenn keine Änderungen der Feldinhalte des Datensatzes vorliegen.

RecNext
Nächster Datensatz
Wert 4 /
0x00000004

Verwandte
Befehle,
Siehe RecRead(),
RecLink(),
RecDelete()

Option bei RecRead(), RecLink() und RecDelete() durch die der nächste (verknüpfte) Datensatz einer Datei (ausgehend vom aktuellen) gelesen/gelöscht werden kann.

_RecNextID
Nächste Datensatz-ID ermitteln
Wert 11

Verwandte

Siehe Befehle,
RecInfo()

Option bei RecInfo() durch die ausgehend von der aktuellen Datensatz-ID die nächste Datensatz-ID der Datei ermittelt werden kann.

_RecNoLoad
Datensatz ohne Feldpufferübertragung lesen

Wert 128 /
0x00000080

Verwandte

Siehe Befehle,
RecRead(),
RecLink()

Option bei RecRead() und RecLink() durch die das Übertragen der Feldpuffer beim Lesen des Datensatzes verhindert werden kann.

Diese Option kann in Verbindung mit _RecLock verwendet werden, um einen Datensatz, der bereits gelesen und in den Feldpuffern geändert wurde, zu sperren. Dabei wird der Datensatz über die Datensatz-ID gesperrt, die aktuell im Puffer steht.


RecPos
Datensatz an bestimmter Schlüssel-/Verknüfungsposition lesen

Wert 5 /
0x00000005

Verwandte

Siehe Befehle,
RecRead(),
RecLink()

Option bei RecRead() und RecLink() durch die der Datensatz an einer bestimmten Schlüssel-/Verknüfungsposition gelesen werden kann.

 Die Schlüsselposition eines Satzes ist bei RecRead() allerdings nur die ungefähre Position, da eine exakte Positionierung zu lange dauern würde (bedingt durch variable Schlüssellängen und dynamisches Speichermanagement in der Datenbank).

RecPrev
Vorheriger Datensatz
Wert 3 /
0x00000003

Verwandte
Befehle,
Siehe RecRead(),
RecLink(),
RecDelete()

Option bei RecRead(), RecLink() und RecDelete() durch die der vorherige (verknüpfte) Datensatz einer Datei (ausgehend vom aktuellen) gelesen/gelöscht werden kann.

_RecSetID
Datensatz-ID setzen
Wert 7

Verwandte

Siehe Befehle,

RecInfo()

Option bei RecInfo() durch die die Datensatz-ID eines Datensatzes oder eines Datensatzpuffers gesetzt werden kann.

RecTest
Datensatzexistenz überprüfen
Wert 64 /
0x00000040

Verwandte
Siehe Befehle,
RecRead(),
RecLink()

Option bei RecRead() und RecLink() durch die die Existenz eines Datensatzes mit einem bestimmten Schlüsselwert überprüft werden kann.

Es findet lediglich ein Zugriff auf den Schlüssel statt, der Datensatz wird dabei nicht gelesen.

RecUpdateCounter
Update-Zähler ermitteln
Wert 13

Verwandte

Siehe Befehle,
RecInfo()

Option bei RecInfo() durch die der Update-Zähler einer Datei ermittelt werden kann. Der Update-Zähler wird beim Öffnen der Datenbank mit 0 initialisiert und bei Verwendung von RecInsert(), RecDelete(), RecDeleteAll() und RecReplace() um 1 erhöht. Durch Datenbank-Reorganisation wird der Zähler wieder auf 0 zurückgesetzt.

RecUseId
Verwendung der Datensatz-ID
Wert 6 /
0x00000006

Verwandte
Siehe Befehle,
RecDelete()

Option bei dem Befehl RecDelete(). Wird diese Option angegeben, wird der Datensatz über die Datensatz-ID gelöscht und nicht über den ersten Schlüssel.

Datensatzzugriff

Vorgehen von CONZEPT 16 beim Lesen eines Datensatzes

RecRead(),

Siehe RecLink(),

SelOpen()

In CONZEPT 16 stehen zwei Befehle zum Lesen eines Datensatzes zur Verfügung. Der Befehl RecRead() liest einen Datensatz aus einer Datei, der Befehl RecLink() liest einen verknüpften Datensatz.

Datensatzzugriff mit RecRead()

Bei dem Befehl RecRead() müssen mindestens drei Parameter übergeben werden. Bei diesen Parametern handelt es sich um die Dateinummer, die Schlüsselnummer und um Leseoptionen. Die Dateinummer bestimmt die Datei, aus der der Datensatz stammt. Anstelle der Dateinummer kann auch ein mit RecBufCreate() angelegter Feldpuffer übergeben werden. Durch die Angabe des Schlüssels werden die Informationen, die zu dem Datensatz bekannt sind definiert.

Beispiel:

Es soll in der Kundendatei (100 / tblCstCustomer) ein Kunde mit dem Namen "vectorsoft AG" gelesen werden. Das Datenbankfeld, welches den Namen der Kunden beinhaltet, ist das Feld "faCstName". Das gleiche Feld ist als Schlüsselfeld in dem Schlüssel keyCstName (dieser Schlüssel hat die Nummer 2) enthalten.

Um einen bestimmten Datensatz zu lesen, wird das Datenbankfeld mit dem gewünschten Wert gefüllt und über einen Schlüssel zugegriffen, der dieses Feld beinhaltet.

```
faCstName # 'vectorsoft AG';tErr # RecRead(100, 2, 0);// besser:tErr # RecRead(tblCstCustomer, ke
```

Das Datenbanksystem ermittelt aus den Informationen, die dem RecRead()-Befehl übergeben werden, das Feld, das den Schlüsselwert enthält. Mit diesem Schlüsselwert wird die Position des Datensatzes in der Datei bestimmt und der Datensatz an dieser Position gelesen. Folgende Fälle können eintreten:

1. Der Datensatz ist vorhanden

Der vollständige Datensatz wird in den Datensatzpuffer geladen. Der Befehl RecRead() gibt bei einem Zugriff über einen eindeutigen Schlüssel als Resultat _rOk zurück. Wurde für den Zugriff ein mehrdeutiger Schlüssel verwendet, kommt _rMultiKey zurück. Der Inhalt des Datensatzes kann über die Namen der entsprechenden Datenbankfelder ausgelesen werden.

2. Der Datensatz ist vorhanden aber von einem anderen Benutzer zur Bearbeitung gesperrt

Der vollständige Datensatz wird in den Datensatzpuffer geladen. Der Befehl RecRead() gibt als Resultat die gleichen Werte wie oben zurück. Der Inhalt des Datensatzes kann über die Namen der entsprechenden Datenbankfelder ausgelesen werden. Wurde beim Lesen die Option _RecLock angegeben, wurde der Datensatz nicht gesperrt und der Wert _rLocked zurückgegeben.

3. Der Datensatz ist nicht vorhanden

In diesem Fall wird der Datensatz an der entsprechenden Position gelesen. Das ist der Datensatz, der in der Schlüsselreihenfolge nach dem gesuchten kommt. Der Befehl RecRead() gibt als Resultat rNoKey zurück.

4. Der Datensatz ist nicht vorhanden und es existiert kein Datensatz an der angegebenen Schlüsselposition

Das kann nur dann eintreten, wenn der gesuchte Datensatz in der Schlüsselreihenfolge nach dem letzten Datensatz in der Datei stehen würde. Es wird der letzte Datensatz gelesen und als Resultat rLastRec zurückgegeben.

5. In der Datei sind keinen Datensätze gespeichert

Dies ist der einzige Fall, in dem kein Datensatz in den Datensatzpuffer geladen wird. Es konnte kein Datensatz gelesen werden. Der Befehl RecRead() gibt als Resultat rNoRec zurück.

Datensatzzugriff mit RecLink()

Mit dem Befehl RecLink() werden, ausgehend von einem gelesenen Datensatz in der Quelldatei, die mit diesem Datensatz verknüpften Datensätze in der Zieldatei gelesen. Es müssen wenigstens vier Parameter übergeben werden: die Dateinummer der Zieldatei, die Dateinummer der Quelldatei, die Verknüpfungsnummer und Leseoptionen.

Der Datensatzzugriff erfolgt mit dem gleichen Prinzip, wie der Zugriff mit dem Befehl RecRead(). Lediglich der Schlüsselwert wird nicht gesetzt, sondern ergibt sich aus dem in der Verknüpfung angegebenen Feld aus der Quelldatei.

Beispiel:

Nachdem der Kundendatensatz der Firma vectorsoft AG gelesen wurde, soll jetzt aus einer verknüpften Datei die erste Telefonnummer gelesen werden. In der verknüpften Datei sind die Telefonnummern zusammen mit dem eindeutigen Schlüsselwert aus der Kundendatei gespeichert. Die Verknüpfung verbindet das Feld mit dem eindeutigen Schlüsselwert aus der Quelldatei mit einem Schlüssel aus der Zieldatei.

```
// Datensatz in der Quelldatei lesenfaCstName # 'vectorsoft AG';tErr # RecRead(tblCstCustomer, ke
```

Der Wert aus der Quelldatei wird zur Positionierung in der Zieldatei verwendet. Der Befehl gibt folgende Werte zurück:

- rOk
Der verknüpfte Datensatz wurde gefunden und in den Datensatzpuffer übertragen.
- rLocked
Der verknüpfte Datensatz wurde gefunden und in den Datensatzpuffer übertragen. Der Datensatz ist aber von einem anderen Benutzer zur Bearbeitung gesperrt.
- rNoRec
Es wurde kein verknüpfter Datensatz gefunden.

Kontakt



Nach dem Lesen eines Datensatzes muss in jedem Fall der Rückgabewert überprüft werden, um festzustellen, ob der gewünschte Datensatz gelesen werden konnte.

Die Datensatz-ID

Eindeutige ID eines Datensatzes

Jeder Datensatz verfügt über eine Datensatz-ID. Die ID ist standardmäßig ein 32-Bit-Wert, der innerhalb einer Datei eindeutig ist. Im Datenstruktureditor können bei Dateien die 64-Bit Datensatz-IDs aktiviert werden.

Beim Einfügen eines neuen Datensatzes in einer Datei wird der Prime-Counter der Datei um 1 erhöht. Anschließend erhält der neue Datensatz den Prime-Counter als Datensatz-ID.



Bei der Datensatz-ID handelt es sich um eine interne ID, die sich durch bestimmte Aktionen (z. B. Export und Import der Datensätze) ändern kann. Sie sollte daher nicht als dauerhafte Referenz auf einen Datensatz verwendet werden.

Die Datensatz-ID eines Datensatzes kann über die Eigenschaft DbRecID ermittelt und über die Anweisung RecInfo() ermittelt oder gesetzt werden. Die zuletzt verwendete Datensatz-ID kann mit der Option _RecGetPrime ermittelt werden.

Die Datensätze erhalten die ID in aufsteigender Reihenfolge beginnend bei 1.

Die Datensatz-ID kann immer dann verwendet werden, wenn ein gelesener Datensatz zu einem späteren Zeitpunkt noch einmal gelesen werden soll. Dies kann zum Beispiel der Fall sein, wenn der Inhalt von Datensätzen durch Intervalle eines Gantt-Diagramm oder durch dynamischen Strukturen dargestellt werden. Wird in den entsprechenden Objekten die Datensatz-ID abgelegt, ist eine einfache Verbindung zum Datensatz geschaffen.

Hat der Prime-Counter einer Tabelle den Maximalwert erreicht, kann kein weiterer Datensatz mehr in die Datei eingefügt werden. Im Datenbank-Log wird der Fehler Prime counter overflow protokolliert. Der Prime-Counter einer Tabelle wird nur zurückgesetzt, wenn sie leer ist und anschließend eine Datenbankdiagnose mit Recover durchgeführt wird.

Datenbankbefehle

Befehle zur Verwaltung einer Datenbank

Siehe [Befehlsgruppen](#),
[Befehlsliste](#)

Befehle

- [DbConnect](#)
- [DbControl](#)
- [DbDisconnect](#)
- [DbInfo](#)
- [DbKeyRebuild](#)
- [DbLicense](#)
- [DbLog](#)
- [DbName](#)

Konstanten

- [_Db?](#)
- [_DbAreaAlias](#)
- [_DbAreaFree](#)
- [_DbAreaName](#)
- [_DbAreaSize](#)
- [_DbAreaVolumeFreeMB](#)
- [_DbAreaVolumeSizeMB](#)
- [_DbBackup](#)
- [_DbBackupRemain](#)
- [_DbBackupStart](#)
- [_DbBackupStop](#)
- [_DbClnLicense](#)
- [_DbClnRelMaj](#)
- [_DbClnRelMin](#)
- [_DbClnRelRev](#)
- [_DbClnRelSub](#)
- [_DbDiagnosis](#)
- [_DbDtaLevel](#)
- [_DbLicenseEndTime](#)
- [_DbLicenseIdnValidThru](#)
- [_DbNoWrites](#)
- [_DbReadOnly](#)
- [_DbRmtProcCount](#)
- [_DbRmtProcLimit](#)
- [_DbSrvHsbIsActive](#)
- [_DbSrvHsbMode](#)
- [_DbSrvLicense](#)
- [_DbSrvRelMaj](#)
- [_DbSrvRelMin](#)
- [_DbSrvRelRev](#)
- [_DbSrvRelSub](#)
- [_DbTimeSync](#)
- [_DbTimeSyncHi](#)
- [_DbUpdate](#)

- DbidUserCount
- DbidUserCountAll
- DbidUserLimit
- DbidUserLimitReal
- KeyNoBreak
- KeyOnlyEmpty
- KeyWait
- LogAlert
- LogError
- LogInfo
- LogWarning

Kontakt

DbConnect(int1, alpha2, alpha3, alpha4, alpha5, alpha6, alpha7)
: int



Verbindung zu einer anderen CONZEPT 16-Datenbank herstellen

int1 Dateinummernbereich

alpha2 Namenspräfix

alpha3 Serveradresse

alpha4 Datenbankname

alpha5 Benutzername

alpha6 Benutzerkennwort

alpha7 Serverkennwort

ErrOk Verbindung erfolgreich
Resultat int hergestellt



 Fehlerwert (siehe Text)

Siehe Verwandte Befehle, DbDisconnect(),
 Verwendung (Blog)

Mit dieser Funktion lassen sich bis zu sieben weitere CONZEPT 16-Datenbanken öffnen. Dabei wird die Datenstruktur einer solchen 'sekundären' Datenbank in einen gesonderten Dateinummernbereich geladen. Dabei stehen sieben Bereiche zur Auswahl (2001-2999, 3001-3999, ... und 8001-8999) die durch Angabe von Db2 bis Db8 in (int1) ausgewählt werden. Das bedeutet, dass beispielsweise die Datei 122 beim Laden in den Bereich 3 dann die Dateinummer 3122 erhält.

Um einen konfliktfreien Zugriff per Namen auf die Datenstrukturen zu ermöglichen, kann in (alpha2) ein Präfix angegeben werden, das allen Elementnamen aus dieser Datenstruktur vorangestellt wird. Das Präfix sollte so gewählt werden, dass keine Überschneidung mit Namen aus der primären Datenstruktur auftritt.

In (alpha3) wird die IP-Adresse oder der Name des Zielserver angegeben. Als Protokoll wird 'TCP' verwendet.

Bei der Verwendung der Hot-Standby-Option des Servers kann der Hostname oder die IP-Adresse der "normalen" Verbindung angegeben werden. In (alpha3) werden beide Server (der Primär- und der Sekundärserver) angegeben. Die beiden Server werden durch '+' voneinander getrennt (<Protokoll>:<Servername>+<Servername>).

Es folgt der Datenbankname in (alpha4) (inklusive Pfadname) bzw. der symbolische Datenbankname.

In den Parametern (alpha5) und (alpha6) werden der Benutzer und sein Kennwort in der verbundenen Datenbank angegeben. Für einen Zugriff via DbConnect() muss die Option "Externer Zugriff" in den Programmrechten des Benutzers der zu verbindenden Datenbank gesetzt sein. Existiert der Benutzer in der zu verbindenden Datenbank nicht oder hat er nicht die erforderlichen Rechte, wird ErrDbUserInvalid zurückgegeben.

Bei kennwortgeschützten Datenbanken kann in (alpha7) das entsprechende Serverkennwort übergeben werden.

Als Resultat wird ein Fehlerwert zurückgeliefert.

Kontakt

<u>ErrOk</u>	Kein Fehler
<u>ErrDbaNoserver</u>	Server nicht vorhanden
<u>ErrDbacomm</u>	Verbindung zum Server unterbrochen
<u>ErrDbanoarea</u>	Datenbank nicht vorhanden
<u>ErrDbareaopen</u>	Datenbank konnte nicht geöffnet werden
<u>ErrDbarealocked</u>	Datenbank ist gesperrt
<u>ErrDbarearollback</u>	Datenbank ist im Rollback-Modus
<u>ErrDbareastandby</u>	Datenbank ist im Standby-Modus
<u>ErrDbareainuse</u>	Die Datenbank ist zur Zeit in exklusiver Benutzung
<u>ErrDbarealockedadmin</u>	Login gesperrt durch den Administrator
<u>ErrDbarealockeddown</u>	Datenbank wird gerade geschlossen
<u>ErrDbarealockednostandbyopen</u>	Zweites Standby-System nicht gestartet
<u>ErrDbarealockedopen</u>	Login-Sperre gesetzt
<u>ErrDbarealockedoperation</u>	Datenbank wird exklusive genutzt
<u>ErrDbarealockedrollback</u>	Rollback-Sperre gesetzt
<u>ErrDbarealockedstandby</u>	Datenbank ist im Standby-Modus
<u>ErrDbareatype</u>	Datenbankversion ungültig
<u>ErrDbareapassword</u>	Serverkennwort nicht korrekt
<u>ErrDbainvalid</u>	Datenbankbenutzer ungültig
<u>ErrDbalimit</u>	Maximale Benutzeranzahl des Servers erreicht
<u>ErrDbaserverstart</u>	Fehler bei Serverstart
<u>ErrDbaserverrelease</u>	Die Server-Version kleiner als die Client-Version



In einer mit DbacConnect() verbundenen Datenbank können nicht alle CONZEPT 16-Befehle verwendet werden. Folgende Befehle stehen zur Verfügung:

- BinDelete
- BinDirDelete
- BinDirOpen
- BinDirRead
- BinExport
- BinImport
- BinOpen
- FileInfo
- FileInfoByName
- FileName
- Fld...
- Fld...ByName
- FldCompare
- FldCopy
- FldDef
- FldDefByName
- FldInfo
- FldInfoByName
- FldName

- KeyFldInfo
- KeyInfo
- KeyInfoByName
- KeyName
- LinkFldInfo
- LinkInfo
- LinkInfoByName
- LinkName
- RecBufClear
- RecBufCompare
- RecBufCopy
- RecBufCreate
- RecBufDefault
- RecDelete
- RecDeleteAll
- RecInfo
- RecInsert
- RecLink
- RecLinkInfo
- RecRead
- RecReplace
- SbrClear
- SbrCompare
- SbrCopy
- SbrInfo
- SbrInfoByName
- SbrName
- SbrStatus
- StoDirOpen
- StoOpen
- TextCopy
- TextCreate
- TextDelete
- TextRead
- TextRename
- TextWrite

Informationen über die verbundene Datenbank können über die Anweisung DbaInfo() durch Angabe der Konstanten _Dba2 bis _Dba8 ermittelt werden.

Texte in der externen Datenbank können über die Konstanten _TextDba2 bis _TextDba8 angesprochen werden.

Beispiele:

```
// Verbindung ohne HSbif (DbaConnect(_Dba3, 'BackUp-', 'TCP:Kronos', 'Kunden', 'SU', '', '') = _E
```

Beim Versuch einen bereits belegten Dateinummernbereich zu verwenden oder bei einem ungültigen Wert in (int1) kommt es zu einem Laufzeitfehler.



Kommt es zu einem Verbindungsabbruch zu einer verbundenen Datenbank, führen die Datensatzbefehle im SOA-Service zum Resultat _ErrDbComm. Beim Standard- und Advanced-Client wird der Client jedoch mit der Meldung "Bereich: Kommunikation / Fehler: Verbindung abgebrochen" beendet. Wird in der Eigenschaft Options des _Sys-Objektes die Option _DbConnectOpErrorCode gesetzt, wird auch im Standard- und Advanced-Client der Fehlercode _ErrDbComm zurückgegeben und der Client bleibt bestehen.

Mögliche Laufzeitfehler:

_ErrValueRange Dateinummernbereich (int1) ungültig

_ErrValueInvalid Dateinummernbereich (int1) bereits belegt

DbacControl(int1[, int2]) : int



CONZEPT 16-Datenbank steuern

Statusfunktion

	<u>DbacBackupStart</u>	Backup-Ereignis starten
	<u>DbacBackupStop</u>	Backup-Ereignis stoppen
int1	<u>DbacTimeSync</u>	Zeitabgleich mit Server durchführen
	<u>DbacTimeSyncHi</u>	Exakten Zeitabgleich mit Server durchführen

int2 Option (optional)

Resultat int ErrOk

Siehe Verwandte Befehle

Mit dieser Funktion können folgende Statusänderungen vorgenommen werden:

- **DbacBackupStart**

Auf dem Server wird ein Backup-Ereignis gestartet. Die Dauer des Ereignisses wird in (int2) in Minuten angegeben. Ein Backup-Ereignis kann bis zu 719 Minuten lang sein. Bis sich die Datenbank im Backup-Modus befindet, können noch mehrere Sekunden vergehen, auch wenn die Funktion sofort zurückkehrt. Wird während eines laufenden Backups ein weiterer Backup gestartet dessen Zeit größer als die Restzeit des laufenden Backups ist, wird der Backup-Modus entsprechend verlängert. Wird ein Backup mit 0 Minuten gestartet, wird ein laufender Backup beendet.

- **DbacBackupStop**

Ein aktives Backup-Ereignis auf dem Server wird gestoppt.

- **DbacTimeSync**

Die aktuelle Uhrzeit des Client-Rechners wird mit der Serverzeit abgeglichen, wobei es zu einer Abweichung bis zu einer Sekunde kommen kann.

- **DbacTimeSyncHi**

Die aktuelle Uhrzeit des Client-Rechners wird mit der Serverzeit exakt abgeglichen.



Hinweise zum Zeitabgleich mit dem Server:

Diese Funktion ist unter UNIX nur mit root-Berechtigung möglich. Unter Windows muss der Benutzer die Berechtigung zum Ändern der Systemzeit besitzen. Um eine korrekte Zeit zu erhalten, muss auf dem Server die richtige Zeitzone eingestellt sein. Dies betrifft auch die Umstellung zwischen Sommerzeit und Normalzeit.

Kontakt

Der Client fordert vom Server die UTC-Zeit (Coordinated Universal Time) an und diese wird je nach Einstellung der Zeitzone auf dem Client-Rechner in die lokale Zeit umgewandelt. Daher ist auch bei den Client-Rechnern auf eine korrekt eingestellte Zeitzone zu achten.

Mögliche Laufzeitfehler:

__ErrNoArgument Backup-Dauer (int2) fehlt (bei Verwendung von __DbBackupStart)
__ErrValueRange Backup-Dauer (int2) ungültig (bei Verwendung von __DbBackupStart)

Db Disconnect(int1)



Verbindung zu einem CONZEPT 16-Server beenden

int1 Dateinummernbereich

Siehe Verwandte Befehle,

Db Connect()

Mit diesem Befehl wird eine Datenbankverbindung, die mit dem Befehl Db Connect() hergestellt wurde, beendet. In (int1) wird der vergebene Nummernbereich (_Db2 bis _Db8) angegeben.

Mögliche Laufzeitfehler:

_ErrValueRange Dateinummernbereich (int1) ungültig

_ErrValueInvalid Dateinummernbereich (int1) nicht belegt



DbInfo(int1[, int2]) : int

Datenbankinformationen ermitteln

int1 Informationstyp (siehe
Text)

int2 Verbundene Datenbank
(optional)

Resultat int Datenbankinformation

Siehe [Verwandte Befehle](#)

Mit dieser Funktion lassen sich folgende Programm- und Datenbankinformationen ermitteln:

- **_DbaDataLevel**

Die Verschachtelungstiefe der Transaktionen des Benutzers. Wird 0 zurückgegeben, ist keine Transaktion geöffnet.

- **_DbUserCount**

Die aktuelle Anzahl der in der Datenbank angemeldeten Benutzer.

- **_DbUserCountAll**

Die aktuelle Anzahl aller Benutzer, die aktuell an dem CONZEPT 16-Server lizenziert sind. Dies sind die Benutzer in allen Datenbanken des Servers. Pro Rechner, der an beliebig vielen Datenbanken des Servers angemeldet ist, wird ein Benutzer lizenziert. Die Anzahl kann mit dem Benutzerlimit (_DbUserLimit) verglichen werden, um festzustellen, von wie vielen Rechnern sich noch Benutzer anmelden können.

- **_DbUserLimit**

Benutzerlimit der Lizenz des CONZEPT 16-Servers.

- **_DbAreaSize**

Die aktuelle Datenbankgröße in KB.

- **_DbAreaFree**

Die Menge des freien Speichers in der Datenbank in KB.

- **_DbAreaVolumeSizeMB**

Größe des Datenträgers, auf dem sich die Datenbank befindet, in MB.

- **_DbAreaVolumeFreeMB**

Größe des freien Platzes auf dem Datenträger, auf dem sich die Datenbank befindet, in MB.

- **_DbReadOnly**

Ist die Datenbank im "Nur-Lesen"-Modus geöffnet, wird 1 zurückgegeben, sonst 0.

- **_DbBackup**

Wird für die Datenbank gerade ein Backup-Ereignis durchgeführt, wird 1 zurückgegeben, sonst 0.

- **_DbBackupRemain**

Die verbleibende Zeit eines Backup-Ereignisses in Sekunden. Der Wert ist 0, wenn kein Backup Backup-Ereignis aktiv ist.

- **_DbUpdate**

Wird für die Datenbank gerade ein Update-Ereignis, d. h. die geänderten Segmente im Datenbankcache werden in die Datenbank geschrieben, durchgeführt, wird 1 zurückgegeben, sonst 0.

- **_DbDiagnosis**

Wird gerade eine Diagnose der Datenbank durchgeführt, wird 1 zurückgegeben, sonst 0.

- **_DbNoWrites**

Kann die Datenbank nicht beschrieben werden (zum Beispiel während eines Backup-Ereignisses, einer Diagnose, der Synchronisation usw.), wird 1 zurückgegeben, sonst 0.

- **_DbSrvHsbMode**

Der Befehl gibt 1 zurück, wenn die Datenbank als Master auf dem Primärsystem geöffnet ist oder die Hot-Standby-Option nicht eingerichtet ist. Das Ergebnis ist 2, wenn die Datenbank auf dem Sekundärsystem als Master geöffnet ist.

- **_DbSrvHsbIsActive**

Der Befehl gibt 1 zurück, wenn die Hot-Standby-Option für die Datenbank eingerichtet und die Datenbank im Hot-Standby-Betrieb (Master-Datenbank ist geöffnet, Slave-Datenbank ist im Standby-Modus) ist. Andernfalls ist das Resultat 0.

- **_DbCacheMB**

Konfigurierte Größe des Datenbankcaches in Megabyte.

- **_DbCacheTempMB**

Größe des für temporäre Daten reservierten Anteils des Datenbankcaches in Megabyte.

- **_DbTempDataKB**

Gesamtmenge der temporären Daten in Kilobyte.

- **_DbClnRelMaj**

Hauptrelease des Clients (5 bei 5.8.01).

- **_DbClnRelMin**

Unterrelease des Clients (8 bei 5.8.01).

- **_DbClnRelRev**

Revision des Clients (1 bei 5.8.01).

- **_DbClnRelSub**

Kontakt

Sub-Revision des Clients (99 bei 5.8.01c). Der Rückgabewert kann mit der Anweisung StrChar() in den Buchstaben gewandelt werden. Gibt es keine Sub-Revisionsnummer wird 0 zurückgegeben.

- **_DbaSrvRelMaj**

Hauptrelease des Servers (5 bei 5.8.01).

- **_DbaSrvRelMin**

Unterrelease des Servers (8 bei 5.8.01).

- **_DbaSrvRelRev**

Revision des Servers (1 bei 5.8.01).

- **_DbaSrvRelSub**

Sub-Revision des Servers (99 bei 5.8.01c). Der Rückgabewert kann mit der Anweisung StrChar() in den Buchstaben gewandelt werden. Gibt es keine Sub-Revisionsnummer wird 0 zurückgegeben.

- **_DbaRmtProcLimit**

Aktuell beim Datenbank-Server eingestellte Prozedur-Limit.

- **_DbaRmtProcCount**

Aktuelle Anzahl in Ausführung befindlicher RmtCall()-Aufrufe.

In (int2) kann angegeben werden, aus welcher verbundenen Datenbank (siehe DbaConnect()) die angegebene Information ermittelt werden soll. Es können die Konstanten _Dba2 bis _Dba8 übergeben werden.

Beispiele:

Datenbankgröße ermitteln:


```
tDbSize # DbaInfo(_DbaAreaSize);
```

Datenbankgröße einer verbundenen Datenbank ermitteln:

```
tDbSize # DbaInfo(_DbaAreaSize,_Dba2);
```

Ermitteln, ob der Primär- oder der Sekundärserver die Datenbank als Master geöffnet hat:

```
if (DbaInfo(_DbaSrvHsbIsActive) = 1 and DbaInfo(_DbaSrvHsbMode) = 1){ // Primary server is Master
```

DbKeyRebuild(int1[, int2,  int3]) : int

Schlüssel reorganisieren

int1 Dateinummer

int2 Schlüsselnummer (optional)

Optionen (optional)

KeyWait Statusanzeige nicht
automatisch
schließen

int3 KeyNoBreak Abbruch nicht
möglich

KeyOnlyEmpty Nur leere Schlüssel
reorganisieren

Reorganisationsresultat

rOk Reorganisation
erfolgreich

rNoKey Kein leerer Schlüssel
vorhanden

Resultat int rUserBreak Abbruch durch
Benutzer

ErrGeneric Schlüsselwertkollision
aufgetreten

Siehe Verwandte Befehle

Mit diesem Befehl wird die Reorganisation von Schlüsseln durchgeführt.

Bei der Schlüsselreorganisation werden alle Datensätze einer Datei gelesen und die Schlüsselwerte erzeugt. Müssen mehr als 10.000 Datensätze reorganisiert werden, parallelisiert der CONZEPT 16-Server diese Aufgabe über mehrere Benutzer. Der Server kann nur dann mehrere Benutzer verwenden, wenn die Anweisung nicht innerhalb einer Transaktion (siehe DtaBegin()) aufgerufen wird. Ob eine Transaktion gestartet wurde, kann mit der Anweisung DbInfo(_DbDtaLevel) ermittelt werden.

Beispiele:

```
// Alle Schlüssel der Datenbank reorganisierenDbKeyRebuild(0);// Alle leeren Schlüssel der Daten
```

Kontakt



Dbalicense(int1) : alpha
Datenbanklizenz ermitteln

	<u>Lizenztyp</u>	
	<u>_DbaclnLicense</u>	Client-Lizenz
	<u>_DbasrvLicense</u>	Server-Lizenz
int1	<u>_DbalicenseEndTime</u>	Ablaufzeitpunkt der Lizenz
	<u>_DbalicenseIdnValidThru</u>	Ablaufzeitpunkt der Lizenz-Identitätsdatei

Resultat alpha Lizenznummer


Siehe Verwandte Befehle

Diese Funktion liefert Lizenzinformationen über Client, Server,
Programmierschnittstelle sowie Web-Schnittstelle.

Es wird für die Clients und den Server die gleiche Lizenznummer zurückgegeben.

Aufgrund der ermittelten Lizenznummern können zum Beispiel Module der
Applikation freigeschaltet oder das Betreiben der Datenbank mit anderen
Lizenznummern unterbunden werden.

Über die Konstante _DbalicenseEndTime kann der Zeitpunkt ermittelt werden, zu
dem eine Lizenz abläuft. Mit der Konstanten _DbalicenseIdnValidThru wird bei der
Verwendung der Lizenz mit Softwareschutz (internetbasierten Lizenz) die Gültigkeit
der Lizenz-Identitätsdatei ermittelt werden. Beide Konstanten geben eine
Zeichenkette zurück, die in den Datentyp bigint und dann in calttime gewandelt
werden kann.

Dbalog(int1, logic2, )
alpha3)

Benutzerlog schreiben

Optionen:

LogInfo Klassifizierung
als Informativer
Eintrag

LogWarning Klassifizierung
als
int1 Warnungseintrag

LogError Klassifizierung
als Fehlereintrag

LogAlert Versand des
Eintrags als
Alert-Mail

logic2 Systemeintrag

alpha3 Text des Eintrags oder
Betreffs (max. 250 Zeichen)

Verwandte Befehle,

Siehe Log-Dateien des
Datenbank-Servers

Mit diesem Befehl wird ein Benutzerlog für die Datenbank geschrieben. Beim ersten Aufrufen dieses Befehls wird im Verzeichnis der Datenbank eine Datei mit dem Namen der Datenbank und der Dateierweiterung lgu angelegt (zum Beispiel CodeLibrary.lgu). In dieser Datei können Log-Einträge durch den Benutzer/Programmierer vorgenommen werden.

Die Benutzereinträge werden in drei Klassen unterschieden, die in (int1) angegeben wird:

- **LogInfo**

Diese Einträge sollten nur informativen Charakter besitzen.

- **LogWarning**

Der Eintrag kennzeichnet einen Warnzustand.

- **LogError**

Der Eintrag kennzeichnet einen Fehlerzustand.

Zusätzlich können die Klassifizierungen LogWarning und LogError mit der Option LogAlert kombiniert werden, um den eingefügten Log-Eintrag als Alert-Mail (siehe automatische E-Mail-Benachrichtigung), an die in AlertMailTo definierte E-Mail-Adresse, zu versenden. Der Betreff für die Alert-Mail lautet "CONZEPT 16 Application alert: <Datenbankname>". Der Versand erfolgt asynchron.

Die unterschiedlichen Klassen werden bei der Anzeige des Protokolls mit dem Log-Viewer durch entsprechende Symbole gekennzeichnet. Neben den übergebenen Parametern wird in der Log-Datei das Datum und die Uhrzeit festgehalten. Mit dem Parameter (logic2) wird definiert, ob es sich bei dem Eintrag um einen Systemeintrag

(true) handelt, oder nicht (false). Systemeinträgen wird in der Log-Datei eine höhere Priorität bei der Langzeitspeicherung gewährt (siehe Benutzerlogs). Die Benutzer-ID wird nur dann eingetragen, wenn es sich bei dem Log-Eintrag nicht um einen System-Eintrag (logic2 = false) handelt.




Das Benutzerlog hat eine Auflösung von 10 Millisekunden. Bei einer zu großen Anzahl an Einträgen verschiebt sich der Zeitpunkt in die Zukunft.

Im Parameter (alpha3) wird der zu speichernde Meldungstext angegeben. Es ist darauf zu achten, dass die Größe der Log-Datei mit der Länge der hier übergebenen Zeilen zusammenhängt und daher bei zu langen Inhalten schnell steigen kann.

Mögliche Laufzeitfehler:

ErrStringOverflow Die in (alpha3) angegebene Zeichenkette war länger als 250 Zeichen.

ErrValueInvalid Der in (int1) übergebene Wert ist nicht gültig.

DbName(int1) : alpha 

Datenbankname ermitteln

Namenstyp

int1 DbAreaName Datenbankname

DbAreaAlias Aliasname der
Datenbank

Resultat alpha Datenbankname

Siehe Verwandte Befehle

Diese Funktion liefert den Namen der geöffneten Datenbank zurück.

Konstanten für Datenbankbefehle
Konstanten für Datenbankbefehle
Siehe Datenbankbefehle

- _Dba?
- _DbaAreaAlias
- _DbaAreaFree
- _DbaAreaName
- _DbaAreaSize
- _DbaAreaVolumeFreeMB
- _DbaAreaVolumeSizeMB
- _DbaBackup
- _DbaBackupRemain
- _DbaBackupStart
- _DbaBackupStop
- _DbaClnLicense
- _DbaClnRelMaj
- _DbaClnRelMin
- _DbaClnRelRev
- _DbaClnRelSub
- _DbaDiagnosis
- _DbaDtaLevel
- _DbaLicenseEndTime
- _DbaLicenseIdnValidThru
- _DbaNoWrites
- _DbaReadOnly
- _DbaRmtProcCount
- _DbaRmtProcLimit
- _DbaSrvHsbIsActive
- _DbaSrvHsbMode
- _DbaSrvLicense
- _DbaSrvRelMaj
- _DbaSrvRelMin
- _DbaSrvRelRev
- _DbaSrvRelSub
- _DbaTimeSync
- _DbaTimeSyncHi
- _DbaUpdate
- _DbaUserCount
- _DbaUserCountAll
- _DbaUserLimit
- _DbaUserLimitReal
- _KeyNoBreak
- _KeyOnlyEmpty
- _KeyWait
- _LogAlert
- _LogError
- _LogInfo
- _LogWarning

_Dba?
Nummernbereich einer verbundenen Datenbank
Wert 2 - 8

DbaConnect()
Siehe DbaDisconnect()
DbaInfo()

Option bei den Befehlen DbaConnect(), DbaDisconnect() und DbaInfo(). Mit den Konstanten wird der Nummernbereich definiert oder die verbundene Datenbank angegeben.

Konstante Nummernbereich

<u>_Db</u> a2	2001-2999
<u>_Db</u> a3	3001-3999
<u>_Db</u> a4	4001-4999
<u>_Db</u> a5	5001-5999
<u>_Db</u> a6	6001-6999
<u>_Db</u> a7	7001-7999
<u>_Db</u> a8	8001-8999

_DbcAreaAlias

Aliasname der Datenbank ermitteln

Wert 1

Verwandte

Siehe Befehle,

DbcName()

Option bei DbcName() durch die der Aliasname der Datenbank ermittelt werden kann.

_DbcAreaName
Datenbankname ermitteln
Wert 0

Verwandte

Siehe Befehle,

DbcName()

Option bei DbcName() durch die der Datenbankname inklusive Verzeichnis ermittelt werden kann.

_DbaBackupStart
Backup-Ereignis starten
Wert 1

Verwandte
Siehe Befehle,
DbaControl(),
Backup-Ereignis

Option bei DbaControl() durch die ein Backup-Ereignis gestartet werden kann.

_DbaBackupStop
Backup-Ereignis beenden
Wert 2

Verwandte
Siehe Befehle,
DbaControl(),
Backup-Ereignis

Option bei DbaControl() durch die ein Backup-Ereignis beendet werden kann.

_DbcClnLicense

Client-Lizenznummer ermitteln

Wert 0

Verwandte

Siehe Befehle,

DbcLicense()

Option bei DbcLicense() durch die die Client-Lizenznummer ermittelt werden kann.

_DbalicenseEndTime
Ablaufzeitpunkt der Lizenz
Wert 2

Verwandte

Siehe Befehle,

Dbalicense()

Option bei Dbalicense(). Wird die Anweisung mit dieser Konstanten aufgerufen, wird der Zeitpunkt zurückgegeben, zu dem die Gültigkeit der Lizenz abläuft. Der zurückgegebene Wert entspricht einem Zeitstempel. Die Zeichenkette kann mit der Anweisung CnvBA() in den Datentyp bigint und anschließend mittels CnvCB() in den Datentyp caltime umgewandelt werden.

Ist die Lizenz zeitlich nicht limitiert, wird der Wert '0' zurückgegeben. Die Anweisung gibt einen Ablaufzeitpunkt zurück, wenn die Datenbank auf dem Sekundärserver einer Hot-Standby-Installation verwendet wird.

_DbalicenseIdnValidThru
Gültigkeit der Lizenz-Identitätsdatei ermitteln
Wert 3

Verwandte

Siehe Befehle,

Dbalicense()

Option bei Dbalicense(). Wird die Anweisung mit dieser Konstanten aufgerufen, wird der Zeitpunkt zurückgegeben, zu dem die Gültigkeit der Lizenz-Identitätsdatei abläuft. Der zurückgegebene Wert entspricht einem Zeitstempel. Die Zeichenkette kann mit der Anweisung CnvBA() in den Datentyp bigint und anschließend mittels CnvCB() in den Datentyp caltime umgewandelt werden.

Wird keine Lizenz mit Softwareschutz (Internetbasierte Lizenz) verwendet oder kann die Lizenz mit Softwareschutz nicht verwendet werden, gibt die Anweisung den Wert '0' zurück.

_DbaSrvLicense

Server-Lizenznummer ermitteln

Wert 1

Verwandte

Siehe Befehle,

Dbalicense()

Option bei Dbalicense() durch die die Server-Lizenznummer ermittelt werden kann.

_DbatimeSync
Zeitabgleich durchführen
Wert 3

Verwandte

Siehe Befehle,

DbacControl()

Option bei DbacControl() durch die ein Zeitabgleich mit dem Server durchgeführt werden kann.

_DbatimeSyncHi
Exakten Zeitabgleich durchführen
Wert 4

Verwandte

Siehe Befehle,

DbatimeSyncHi

Option bei DbatimeSyncHi durch die ein exakter Zeitabgleich mit dem Server durchgeführt werden kann.

Während bei DbatimeSync die Abweichung zwischen Client und Server immer noch bis zu 1 Sekunde betragen kann, liegt die Abweichung bei _DbatimeSyncHi meist unter 10ms. Der Aufruf von DbatimeSyncHi kann dabei allerdings bis zu 1 Sekunde lang dauern.

_KeyNoBreak
Abbruch nicht möglich
Wert 2 / 0x00000002

Verwandte

Siehe Befehle,

DbKeyRebuild()

Option bei DbKeyRebuild() durch die ein Abbruch der Schlüsselreorganisation verhindert werden kann.

_KeyOnlyEmpty
Nur leere Schlüssel reorganisieren
Wert 4 / 0x00000004

Verwandte

Siehe Befehle,

DbKeyRebuild()

Option bei DbKeyRebuild() durch die die Schlüsselreorganisation auf leere Schlüssel beschränkt werden kann.

_KeyWait

Statusanzeige nicht automatisch schließen

Wert 1 / 0x00000001

Verwandte

Siehe Befehle,

DbKeyRebuild()

Option bei DbKeyRebuild() durch die ein automatisches Schließen der Statusanzeige nach der Schlüsselreorganisation verhindert werden kann.

_LogAlert

Eintrag als Alert-Mail versenden

Wert 16 / 0x10

Dbalog(),

Siehe _LogWarning,

_LogError

Übergabeparameter bei der Anweisung Dbalog() mit der der aktuelle Logeintrag als Alert-Mail (siehe automatische E-Mail-Benachrichtigung) versendet wird. Dies ist nur für die Klassen _LogWarning und _LogError möglich. Der Versand erfolgt asynchron.

_LogError

Fehlereintrag im Benutzerlog

Wert 4

Siehe Dbalog()

Übergabeparameter bei der Anweisung Dbalog() - Der Eintrag in das Benutzerprotokoll kennzeichnet einen Fehlerzustand.

_LogInfo

Informativer Eintrag im Benutzerlog

Wert 1

Siehe Dbalog()

Übergabeparameter bei der Anweisung Dbalog() - Der Eintrag in das Benutzerprotokoll ist nur informativer Art.

_LogWarning

Warneintrag im Benutzerlog

Wert 3

Siehe Dbalog()

Übergabeparameter bei der Anweisung Dbalog() - Der Eintrag in das Benutzerprotokoll kennzeichnet einen Warnzustand.

Befehle für binäre Objekte

Liste der Befehle und Konstanten zur Bearbeitung von binären Objekten

Befehlsgruppen,

Siehe Befehlsliste,

Binäre Objekte

Befehle

- BinClose
- BinCopy
- BinDelete
- BinDirDelete
- BinDirMove
- BinDirOpen
- BinDirRead
- BinExport
- BinImport
- BinMove
- BinOpen
- BinReadMem
- BinRename
- BinUpdate
- BinWriteMem

Konstanten

- _BinClearOnly
- _BinCreate
- _BinCreateNew
- _BinDbg?
- _BinDeleteAll
- _BinDirectory
- _BinErrorDecryption
- _BinFirst
- _BinLast
- _BinLock
- _BinNext
- _BinPrev
- _BinSharedLock
- _BinSingleLock

obj -> BinClose()



Verzeichnis/Objekt schließen

obj Verzeichnis/Objekt

Verwandte

Siehe Befehle,

BinOpen(),

BinDirOpen()

Mit dieser Funktion wird das Verzeichnis oder das binäre Objekt geschlossen und entsperrt. Der Deskriptor ist anschließend nicht mehr gültig.



Beim Bearbeiten binärer Objekte in einer mit DbConnect() verbundenen Datenbank ist zu beachten, dass die Funktion DbDisconnect() geöffnete Verzeichnisse und binäre Objekte schließt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Verzeichnis/Objekt (obj) ungültig

obj -> BinCopy(handle1) :

int

Objektinhalt kopieren

obj Ausgangsobjekt

handle1 Zielobjekt

Kopierresultat

ErrOk

Kopieren
erfolgreich

ErrBinNoLock

Zielobjekt
(int1) nicht
gesperrt

ErrBinOperation

Resultat int Versuch
Objekt in eine

mit
DbConnect()
verbundene
Datenbank zu
kopieren

rDeadlock



Verklemmung
aufgetreten

Siehe Verwandte Befehle, BinMove()

Mit dieser Funktion wird der Inhalt des Ausgangsobjekts (obj) in das Zielobjekt (handle1) übertragen. Das Zielobjekt muss dazu exklusiv gesperrt sein (siehe BinLock und BinSingleLock).


Mögliche Laufzeitfehler:

ErrHdlInvalid Ausgangsobjekt (obj) ungültig

obj -> BinDelete(alpha1[, 
int2]) : int
Objekt löschen/leeren
obj Ausgangsverzeichnis
alpha1 Objektname
Optionen (optional)
_BinClearOnly Objektinhalt löschen
_BinDirectory Verzeichnis löschen
int2 _BinDeleteAll Verzeichnis und
Unterverzeichnisse
löschen
_BinDbas? Datenbankbereich
Löschresultat
_ErrOk Löschen
erfolgreich
_ErrBinNameInvalid Objektname
(alpha1)
ungültig
_ErrBinNoFile Objekt in
Objektname
(alpha1)
existiert nicht
Resultat int _ErrBinNoPath Verzeichnis 
in
Objektname
(alpha1)
existiert nicht
_ErrBinLocked Objekt
(alpha1)
gesperrt
_ErrDeadlock Verklemmung
aufgetreten

Siehe Verwandte Befehle, BinOpen()

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

 Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (obj) 0 angegeben, und dieses vor den Befehlsargumenten aufgeführt: BinDelete(0, alpha1[, int2])

Folgende Optionen (int2) können angegeben werden:

- _BinClearOnly

Der Inhalt des Objekts (alpha1) wird gelöscht.

- _BinDbas?

Das Objekt wird in einer mit DbasConnect() verbundenen Datenbank gelöscht oder geleert. Der Datenbankbereich wird in der Option mit _BinDbas2 bis

- BinDbas angegeben.
- BinDirectory

Das Verzeichnis (alpha1) wird gelöscht. (Siehe BinDirDelete())

- BinDeleteAll

Das Verzeichnis (alpha1) alle Unterverzeichnisse und enthaltenen Objekte werden gelöscht. Die Option BinDirectory muss angegeben sein.

Beispiele:

```
// Objekt 'File' im Verzeichnis tHdl löscht tHdl->BinDelete('File');// Objekt 'Tab4' im Verzeichnis tHdl löscht tHdl->BinDelete('Tab4');
```


Mögliche Laufzeitfehler:

ErrHdlInvalid Ausgangsverzeichnis (obj) ungültig

obj -> BinDirDelete(alpha1[,
int2]) : int
Binäres Verzeichnis löschen
obj Ausgangsverzeichnis
alpha1 Verzeichnisname
Optionen (optional)
int2 BinDeleteAll Verzeichnis leeren und
löschen
BinDbg? Datenbankbereich
Löschresultat
ErrOk Löschen
erfolgreich
ErrBinNameInvalid Verzeichnisname
(alpha1) ungültig
ErrBinNoPath Verzeichnispfad
in
Verzeichnisname
Resultat int (alpha1) ungültig
ErrBinLocked Verzeichnis
(alpha1) gesperrt
ErrBinDirNotEmpty Verzeichnis nicht
leer, oder enthält
geöffnete Objekte
rDeadlock Verklemmung
aufgetreten

Siehe Verwandte Befehle, BinDirOpen(),
BinDelete()

Mit dieser Funktion wird das Verzeichnis (alpha1) im Ausgangsverzeichnis (obj)
gelöscht.

 Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (obj)
0 angegeben, und dieses vor den Befehlsargumenten aufgeführt: BinDirDelete(0,
alpha1[, int2])

Der Verzeichnisname in (alpha1) kann auch einen Pfadbestandteil enthalten, der
relativ zum Ausgangsverzeichnis ist.

Beispiele:

```
// Verzeichnis 'Test' löschenBinDirDelete(0, 'Test');// Unterverzeichnis 'Documents\Excel' von th
```

Folgende Optionen (int2) können verwendet werden:

- BinDeleteAll

Das Verzeichnis (alpha1) und alle darin enthaltenen Unterverzeichnisse und
Objekte werden gelöscht.

- BinDbg?

Kontakt

Das Verzeichnis wird in einer mit DbConnect() verbundenen Datenbank gelöscht. Der Datenbankbereich wird in der Option mit _BinDb2 bis _BinDb8 angegeben.

Mögliche Laufzeitfehler:

_ErrHdlInvalid Ausgangsverzeichnis (obj) ungültig

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



Binäres Verzeichnis verschieben

obj Zu verschiebendes Verzeichnis

handle1 Zielverzeichnis oder 0

int2 Option (optional)

BinDbas? Datenbankbereich

Verschieberesultat

ErrOk Verschieben
erfolgreich oder
Verzeichnis
bereits im
Zielverzeichnis

ErrBinNoLock Verzeichnis (obj)
nicht gesperrt

ErrBinExists Verzeichnis (obj)
existiert bereits
im
Zielverzeichnis
(handle1)

Resultat int ErrBinOperation Versuch
Verzeichnis
zwischen zwei
Datenbanken
(siehe
DbasConnect(),
Verzeichnis in
sich selbst, oder
in ein sich
untergeordnetes
Verzeichnis zu
verschieben



rDeadlock Verklemmung
aufgetreten

Siehe Verwandte Befehle, BinMove()

Mit dieser Funktion wird das binäre Verzeichnis (obj) in das Zielverzeichnis (handle1) verschoben. Das Objekt muss dazu exklusiv gesperrt sein (siehe BinLock und BinSingleLock). Das Wurzelverzeichnis kann nicht verschoben werden.

Um ein Verzeichnis in das Wurzelverzeichnis zu verschieben, wird als Zielverzeichnis (handle1) 0 angegeben.

Soll ein binäres Verzeichnis in einer mit DbasConnect() verbundenen Datenbank in das Wurzelverzeichnis der verbundenen Datenbank verschoben werden, muss als Option (int2) die entsprechende BinDbas?-Konstante angegeben werden.




Binäre Verzeichnisse können nicht zwischen zwei Datenbanken verschoben werden.

Mögliche Laufzeitfehler:

Kontakt

ErrHdlInvalid Zu verschiebendes Verzeichnis (obj) oder Zielverzeichnis (handle1)
ungültig.

Kontakt

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

handle
Verzeichnis öffnen/erstellen

obj Ausgangsverzeichnis
alpha1 Verzeichnisname
Optionen (optional)

_BinLock Verzeichnis für andere Benutzer sperren

_BinSharedLock Verzeichnis mit anderen Benutzern sperren

int2 _BinSingleLock Verzeichnis für alle Benutzer sperren

_BinCreate Verzeichnis erstellen


_BinCreateNew Verzeichnis explizit erstellen

_BinDbg? Datenbankbereich

Öffnungs-/Anlegeresultat
Verzeichnis-Deskriptor
oder

_ErrBinNameInvalid Verzeichnisname (alpha1) ungültig

_ErrBinNoFile Verzeichnis (alpha1) unbekannt

_ErrBinNoPath Verzeichnis in Verzeichnispfad (alpha1) unbekannt 

Resultat handle _ErrBinLocked Verzeichnis (alpha1) gesperrt

_ErrBinExists Verzeichnis (alpha1) existiert bereits bei _BinCreateNew

_ErrBinOperation Maximale Ebenenanzahl von 60 überschritten

_rDeadlock Verklemmung aufgetreten

Siehe Verwandte Befehle, BinClose(), Binäre Objekte

Mit dieser Funktion wird ein Verzeichnis von binären Objekten geöffnet oder neu angelegt. In (obj) wird der Deskriptor des Ausgangsverzeichnis angegeben.



Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (obj) 0 angegeben, und dieses vor den Befehlsargumenten aufgeführt. Das

Wurzelverzeichnis ist immer vorhanden und braucht auch nicht geöffnet zu werden: `BinDirOpen(0, alpha1[, int2])`

Die maximale Länge eines Verzeichnisnamens (ohne Pfad) beträgt 60 Zeichen. Der Verzeichnisname darf keine Steuerzeichen oder die Zeichen * und ? enthalten. Es können maximal 60 Ebenen angelegt werden.

Der Verzeichnisname (alpha1) kann auch einen Pfadb Bestandteil enthalten, der relativ zum Ausgangsverzeichnis ist.

Folgende Optionen (int2) können angegeben werden:

- BinCreate

Das Verzeichnis wird im Ausgangsverzeichnis erstellt.

- BinDbas?

Das Verzeichnis wird in einer mit DbasConnect() verbundenen Datenbank geöffnet oder erstellt. Der Datenbankbereich wird in der Option mit BinDbas2 bis BinDbas8 angegeben.

- BinLock

Das Verzeichnis wird beim Öffnen oder Anlegen für andere Benutzer gesperrt.

- BinSharedLock

Das Verzeichnis wird beim Öffnen oder Anlegen mit anderen Benutzer gesperrt.

- BinSingleLock

Das Verzeichnis wird beim Öffnen oder Anlegen für alle Benutzer gesperrt.



Wird keine Sperroption angegeben, wird das Verzeichnis mit einer gemeinsamen Sperre (siehe BinSharedLock) geöffnet.

Die Sperrung eines Verzeichnisses bleibt bis zum Schließen des Verzeichnisses mit BinClose() oder bis sich der Benutzer von der Datenbank abmeldet erhalten.

Änderungen an einem Verzeichnis (Update, Import usw.) können nur bei einer exklusiven Sperre (siehe BinLock und BinSingleLock) vorgenommen werden.

Beispiele:

```
// Verzeichnis 'Test' im Wurzelverzeichnis öffnetBinObj # BinDirOpen(0, 'Test');// Verzeichnispt
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Ausgangsverzeichnis (obj) ungültig

obj ->

BinDirRead(int1[,
alpha2]) : alpha



Verzeichnis lesen

obj Ausgangsverzeichnis

Lesemodus

0 Das angegebene
Objekt lesen

_BinFirst Ersten Eintrag
lesen

_BinLast Letzten Eintrag
lesen

int1 _BinNext Eintrag nach
Referenzeintrag
lesen

_BinPrev Eintrag vor
Referenzeintrag
lesen

_BinDirectory Unterverzeichnisse
lesen

_BinDbas? Datenbankbereich

alpha2 Referenzeintrag (optional)

Resultat alpha Eintragsname

Siehe Verwandte Befehle,
BinDirOpen()

Diese Funktion liest den Namen eines Eintrags aus dem Ausgangsverzeichnis (obj).
Das Ausgangsverzeichnis (obj) muss mit BinDirOpen() geöffnet worden sein.



Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (obj) 0 angegeben, und dieses vor den Befehlsargumenten aufgeführt: BinDirRead(0, int1, int2[, alpha3]).

Das Lesen der Verzeichniseinträge kann über folgende Optionen (int1) erfolgen:

- **0**

Der Verzeichniseintrag mit dem in (alpha2) angegebenem Namen wird gelesen. Ist kein Verzeichniseintrag mit dem Namen vorhanden, wird der Eintrag mit dem nächst höheren Namen gelesen. Ist kein nächst höherer vorhanden, wird ein Leerstring zurückgegeben.

- **_BinFirst**

Der erste Verzeichniseintrag wird gelesen.

- **_BinLast**

Der letzte Verzeichniseintrag wird gelesen.

- **_BinNext**

Der Verzeichniseintrag nach dem Referenzeintrag (alpha2) wird gelesen.

- **_BinPrev**

Kontakt

Der Verzeichniseintrag vor dem Referenzeintrag (alpha2) wird gelesen.

- **BinDirectory**

Die Unterverzeichnisse werden gelesen.

- **BinDbas?**

Das Verzeichnis wird in einer mit DbasConnect() verbundenen Datenbank gelesen. Der Datenbankbereich wird in der Option mit BinDbas2 bis BinDbas8 angegeben. Diese Option wirkt sich nur auf das Wurzelverzeichnis (obj = 0) aus.

Konnte kein Eintrag gelesen werden (zum Beispiel, weil bei BinNext kein Folgeeintrag existiert), wird als Ergebnis eine leere Zeichenkette zurückgegeben (").


Beispiel:

```
// Verzeichnis 'Test' öffnentHdl # BinDirOpen(0, 'Test');// Verzeichnis vorhandenif (tHdl > 0){
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Ausgangsverzeichnis (obj) ungültig

obj -> BinExport(alpha1[,
alpha2[, int3]]) : int
Objekt exportieren
obj Objekt
alpha1 Name der externen Datei
alpha2 Verschlüsselungs-Code (optional)
Optionen (optional)
 BinErrorDecryption Eindeutiger Fehlerwert
 wenn
 Entschlüsselungscode
 falsch
int3 FsiNameC16 Dateiname/-pfad (alpha1)
 wird im
 CONZEPT 16-Zeichensatz
 erwartet (Standard)
 FsiNameUtf8 Dateiname/-pfad (alpha1)
 wird im
 UTF-8-Zeichensatz
 erwartet

Resultat int Exportresultat (siehe Text) 

Siehe Verwandte Befehle, BinReadMem(),
 BinOpen(), BinImport()

Mit dieser Funktion wird der Inhalt des Objektes (obj) in die externe Datei (alpha1) exportiert. Falls der Objektinhalt verschlüsselt gespeichert wurde, muss in (alpha2) der entsprechende Verschlüsselungscode angegeben werden. Bei einem inkorrekten Code ist das Resultat ErrBinData. Falls das Objekt leer ist, wird ErrBinNoData zurückgeliefert.

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

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

Optional können folgende Optionen (int3) angegeben werden:

BinErrorDecryption Wird ein falscher Entschlüsselungscode angegeben, wird ErrBinDecryption, statt dem allgemeinen Fehlerwert, ErrBinData, zurückgegeben.
FsiNameC16 Der Dateiname/-pfad (alpha1) wird im CONZEPT 16-Zeichensatz erwartet.
FsiNameUtf8 Der Dateiname/-pfad (alpha1) wird im UTF-8-Zeichensatz erwartet. Somit ist es möglich, binäre Objekte mit Umlauten anderer Sprachen zu exportieren.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:


ErrBinData Verschlüsselungs-Code (alpha2) falsch (wenn Option (int3) BinErrorDecryption nicht gesetzt) oder allgemeiner Fehler

Kontakt

<u>ErrBinDecryption</u>	Verschlüsselungs-Code (alpha2) falsch (wenn Option (int3) <u>BinErrorDecryption</u> gesetzt)
<u>ErrBinNoData</u>	Objekt (obj) leer
<u>ErrFsiNoPath</u>	Pfad im Namen der externe 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

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Objekt (obj) ungültig
<u>ErrValueInvalid</u>	Es wurde eine ungültige Option (int3) angegeben.

obj -> BinImport(alpha1[, 
int2[, alpha3[, int4]]]) : int

Objekt importieren

obj Objekt

alpha1 Pfad und Name einer externen Datei


int2 Kompressionsstufe (optional)

alpha3 Verschlüsselungs-Code (optional)

Optionen (optional)

FsiNameC16 Dateiname/-pfad (alpha1)
wird im
CONZEPT 16-Zeichensatz
erwartet (Standard)

int4 FsiNameUtf8 Dateiname/-pfad (alpha1)
wird im
UTF-8-Zeichensatz
erwartet

Resultat int Importresultat (siehe Text) 

Siehe Verwandte Befehle, BinOpen(),
BinExport()

Mit dieser Funktion wird der Inhalt der externen Datei (alpha1) in das Objekt (obj) importiert. Ein bereits bestehender Inhalt wird dabei überschrieben. Das Objekt muss dazu exklusiv gesperrt sein (siehe BinLock und BinSingleLock). Die externe Datei darf nicht leer sein.

Optional kann der Inhalt mit den Stufen 1 bis 4 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 (alpha3) übergeben (siehe StrEncrypt()). Es ist zu beachten, dass ohne diesen Code der Objekthinhalte nicht mehr gelesen werden kann.

Wird im optionalen Argument (int5) FsiNameUtf8 angegeben, wird der Dateiname/-pfad (alpha1) als UTF-8-Zeichenkette erwartet. Somit ist es auch möglich Dateien mit Umlauten anderer Sprachen zu importieren.

Folgende Fehlerwerte werden von der Funktion zurückgegeben:


<u>ErrBinNoLock</u>	Objekt (obj) nicht gesperrt
<u>ErrBinNoData</u>	Externe Datei (alpha1) leer (0 Byte)
<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

Kontakt


_ErrFsiSharingViolation Zugriffskonflikt bei Zugriff auf externe Datei (alpha1)
_ErrFsiLockViolation Sperrkonflikt bei Zugriff auf externe Datei (alpha1)
_ErrFsiOpenFailed Externe Datei (alpha1) konnte nicht geöffnet werden
_rDeadlock Verklemmung aufgetreten

Mögliche Laufzeitfehler:

_ErrHdlInvalid Objekt (obj) ungültig
_ErrValueInvalid Es wurde eine ungültige Option (int5) angegeben.

obj -> BinMove(handle1) 
 : int
 Objekt verschieben
 obj Objekt
 handle1 Zielverzeichnis

Verschieberesultat

<u>_ErrOk</u>	Verschieben erfolgreich
<u>_ErrBinNoLock</u>	Objekt (obj) nicht gesperrt
<u>_ErrBinExists</u>	Objekt (obj) existiert bereits im Zielverzeichnis (handle1) 

Resultat int

<u>_ErrBinOperation</u>	Versuch Objekt in eine mit <u>DbConnect()</u> verbundene Datenbank zu verschieben
<u>_rDeadlock</u>	Verklemmung aufgetreten


Siehe Verwandte Befehle, BinCopy(),
BinDirMove()

Mit dieser Funktion wird das Objekt (obj) in das Zielverzeichnis (handle1) verschoben.
 Das Objekt muss dazu exklusiv gesperrt sein (siehe _BinLock und _BinSingleLock).
 Binäre Objekte können nicht in das Wurzelverzeichnis verschoben werden.

Mögliche Laufzeitfehler:

_ErrHdlInvalid Objekt (obj) oder Zielverzeichnis (handle1) ungültig

Kontakt

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

handle
Objekt öffnen/erstellen
obj Ausgangsverzeichnis
alpha1 Objektname
Optionen (optional)

- __BinLock Objekt für andere Benutzer sperren
- __BinSharedLock Objekt mit anderen Benutzern sperren
- int2 __BinSingleLock Objekt für alle Benutzer sperren
- __BinCreate Objekt erstellen
- __BinCreateNew Objekt explizit erstellen
- __BinDirectory Verzeichnis öffnen
- __BinDbas? Datenbankbereich


Öffnungs-/Anlegeresultat
Objekt-Deskriptor
oder

- __ErrBinNameInvalid Objektname (alpha1) ungültig
- __ErrBinNoPath Objekt in Objektpfad (alpha1) ungültig
- __ErrBinNoFile Objekt (alpha1) unbekannt
- __ErrBinLocked Objekt (alpha1) gesperrt
- __ErrBinExists In einem zweiten Client wurde ein binäres Objekt mit dem gleichen Namen angelegt und noch nicht mit __BinClose() geschlossen oder bei __BinCreateNew existierte das binäre Objekt bereits
- __ErrBinOperation Versuch Objekt im Wurzelverzeichnis zu erstellen
- __rDeadlock Verklemmung aufgetreten

Resultat handle

Siehe Verwandte Befehle, __BinClose(), Binäre Objekte

Mit dieser Funktion wird ein binäres Objekt geöffnet oder neu angelegt. In (obj) wird der Deskriptor des Ausgangsverzeichnisses angegeben.

-  Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (obj) 0 angegeben, und dieses vor den Befehlsargumenten aufgeführt: `BinOpen(0, alpha1[, int2])`

Die maximale Länge eines Objektnamens (ohne Pfad) beträgt 60 Zeichen. Der Objektname darf keine Steuerzeichen oder die Zeichen * und ? enthalten.

Der Objektname (alpha1) kann auch einen Pfadbestandteil enthalten, der relativ zum Ausgangsverzeichnis ist.

Folgende Optionen (int2) können angegeben werden:

- BinCreate

Das Objekt wird im Ausgangsverzeichnis erstellt. Binäre Objekte können nicht im Wurzelverzeichnis erstellt werden.

- BinCreateNew

Das Objekt wird im Ausgangsverzeichnis explizit erstellt. Binäre Objekte können nicht im Wurzelverzeichnis erstellt werden. Existiert das Objekt bereits, wird der Fehlerwert ErrBinExists zurückgegeben.

- BinDbas?

Das Objekt wird in einer mit DbasConnect() verbundenen Datenbank geöffnet oder erstellt. Der Datenbankbereich wird in der Option mit BinDbas2 bis BinDbas8 angegeben.

- BinDirectory

Es wird ein Verzeichnis geöffnet. (Siehe BinDirOpen())

- BinLock


Das Objekt wird beim Öffnen oder Anlegen für andere Benutzer gesperrt.

- BinSharedLock

Das Objekt wird beim Öffnen oder Anlegen mit anderen Benutzer gesperrt.

- BinSingleLock

Das Objekt wird beim Öffnen oder Anlegen für alle Benutzer gesperrt.

-  Wird keine Sperroption angegeben, wird das Objekt mit einer gemeinsamen Sperre (siehe BinSharedLock) geöffnet.


Die Sperrung eines Objektes bleibt bis zum Schließen des Objektes mit BinClose() oder bis sich der Benutzer von der Datenbank abmeldet erhalten. Änderungen an einem Objekt (Update, Import usw.) können nur bei einer exklusiven Sperre (siehe BinLock und BinSingleLock) vorgenommen werden.

Beispiele:

```
// Objekt 'Tab1' im Verzeichnis 'Test\Dokumente\Excel' öffnetBinObj # BinOpen(0, '\\Test\Dokumente\Excel\Tab1')
```

Mögliche Laufzeitfehler:

_ErrHdlInvalid Ausgangsverzeichnis (obj) ungültig

obj -> BinReadMem(handle1[, alpha2[,  int3]]) : int

Binäres Objekt in Memory-Objekt lesen

obj Deskriptor eines binären Objekts

handle1 Deskriptor eines Memory-Objekts

alpha2 Verschlüsselungscode (optional)

Optionen (optional)

int3 __BinErrorDecryption Eindeutiger
Fehlerwert wenn
Entschlüsselungscode
falsch

Resultat int Fehlerwert 

Siehe Verwandte Befehle, BinExport(),
BinWriteMem()

Mit dieser Funktion wird der Inhalt des binären Objekts (obj) in das Memory-Objekt (handle1) eingelesen. Falls der Objekthalt verschlüsselt gespeichert wurde, muss in (alpha2) der entsprechende Verschlüsselungscode angegeben werden. Bei einem inkorrekten Code ist das Resultat __ErrBinData. Falls das Objekt leer ist, wird __ErrBinNoData zurückgeliefert. In allen anderen Fällen ist das Resultat __ErrOk.


Der Wert der Eigenschaft Len entspricht nach der Operation der unkomprimierten Datengröße des binären Objekts.

Optional kann als Option (int3) __BinErrorDecryption angegeben werden um bei einem falschen Entschlüsselungscode __ErrBinDecryption statt dem allgemeinen Fehlerwert, __ErrBinData, zu erhalten.

Mögliche Laufzeitfehler:

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

__ErrStringOverflow Das zu lesenden binäre Objekt ist größer als das Memory-Objekt.

obj -> BinRename(alpha1) : int 


Verzeichnis/Objekt umbenennen

obj Objekt/Verzeichnis

alpha1 Neuer Name

Umbenennungsresultat

ErrOk Verschieben
erfolgreich

ErrBinNoLock Objekt/Verzeichnis
(obj) nicht
gesperrt 

Resultat int

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

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

: int


Binäres Objekt aus Memory-Objekt schreiben

obj Deskriptor eines
 binären Objekts

handle1 Deskriptor eines
 Memory-Objekts

int2 Kompressionsstufe
 (optional)

alpha3 Verschlüsselungs-Code
 (optional)

Resultat int Fehlerwert 

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 Objekthinhalte 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 Deskriptor 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

Siehe für
binäre
Objekte

- _BinClearOnly
- _BinCreate
- _BinCreateNew
- _BinDbg?
- _BinDeleteAll
- _BinDirectory
- _BinErrorDecryption
- _BinFirst
- _BinLast
- _BinLock
- _BinNext
- _BinPrev
- _BinSharedLock
- _BinSingleLock

_BinClearOnly
Objekthalt 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.

_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.

_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.

_BinFirst
Ersten Eintrag lesen
Wert 1 /
0x00000001

Verwandte

Siehe Befehle,

BinDirRead()

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

_BinLast
Letzten Eintrag lesen
Wert 2 /
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 8 /
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 4 /
0x00000004

Verwandte

Siehe Befehle,

BinDirRead()

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

BinPrev
Eintrag vor Referenzeintrag lesen
Wert 3 /
0x00000003

Verwandte

Siehe Befehle,

BinDirRead()

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

_BinSharedLock
Objekt/Verzeichnis gemeinsam mit anderen Benutzer sperren
Wert 48 /
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 40 /
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.

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.

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

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 löscht tHdl->StoDelete('File');// Objekt 'File' in verbunden
```

Mögliche Laufzeitfehler:

Kontakt

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 tName
```

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

handle

Öffnen eines Storage-Verzeichnisses

obj Eltern-Verzeichnis

alpha1 Name des Verzeichnis

Optionen (optional)

int2 StoDb? 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 Themes

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

Als optionaler Parameter (int2) kann eine der Konstanten StoDb2 bis StoDb8 angegeben werden, wenn Storage-Verzeichnisse aus einer mit Dbconnect() 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.

obj ->
 StoExport(alpha1)       
 : int
 Objekt exportieren
 obj Objekt
 alpha1 Name der externen
 Datei
 Resultat int Exportresultat 
 (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

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,

Siehe StoWriteTileMem(), StoImport(),
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.

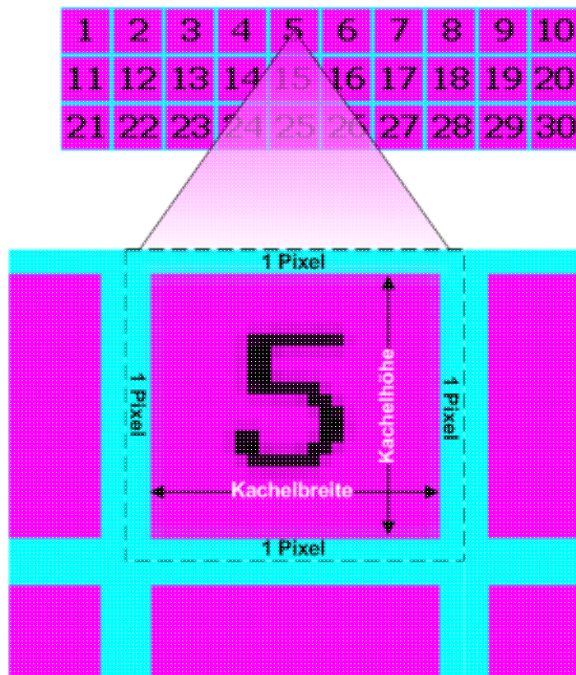
Bei Bildern mit Alpha-Kanal (32 Bit Farbtiefe) werden diese Farben nicht berücksichtigt.

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

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

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 worden 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



obj -> StoReadMem(handle1) : int
Storage-Objekt in Memory-Objekt lesen

obj Deskriptor eines
 Storage-Objektes

handle1 Deskriptor eines
 Memory-Objektes

Resultat int Fehlerwert

Verwandte

Siehe Befehle,
 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.



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.

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.

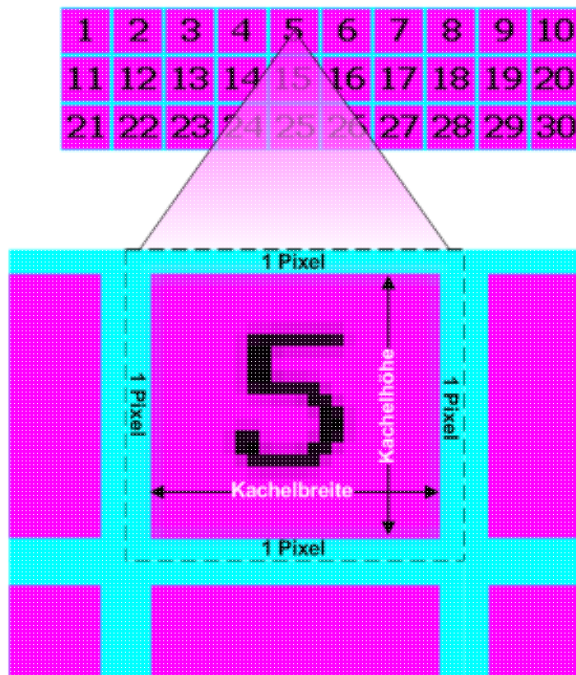
Bei Bildern mit Alpha-Kanal (32 Bit Farbtiefe) werden diese Farben nicht berücksichtigt.

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

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</u> -Objekt enthält keine Daten
<u>ErrStoLocked</u>	Das <u>Storage</u> -Objekt 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>	Verklemmung aufgetreten

Mögliche Laufzeitfehler:

<u>ErrHdllInvalid</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

_StoCreate
Storage-Objekt erstellen
Wert 4.096 /
0x00001000

Verwandte

Siehe Befehle,
StoOpen()

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

_StoDba?

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 : _StoDba2

Nummernbereich 3 : _StoDba3

Nummernbereich 4 : _StoDba4

Nummernbereich 5 : _StoDba5

Nummernbereich 6 : _StoDba6

Nummernbereich 7 : _StoDba7

Nummernbereich 8 : _StoDba8

_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().

_StoFirst
Erstes Storage-Objekt lesen
Wert 1 /
0x00000001

Siehe StoDirRead()

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

_StoPrev
Vorhergehendes Verzeichnis lesen
Wert 3 /
0x00000003

Siehe StoDirRead()

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

_StoNext
Nächstes Verzeichnis lesen
Wert 4 /
0x00000004

Siehe StoDirRead()

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

_StoLast
Letztes Verzeichnis lesen
Wert 2 /
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,

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

Bedingungen

Alle einschränkenden Felder befinden sich in einem Schlüssel.

Es wird nur auf einen Schlüsselwert eingeschränkt.

Alle einschränkenden Felder befinden sich in einem Schlüssel.

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

Die Bedingungen müssen nicht geklammert werden.

-

-> **Objekt**

-> Verknüpfung

-> Filter

-> 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

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

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,
```

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

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.

 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 Datensätzen 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:

<u>SelServer</u>	Ausführung auf dem Server
<u>SelServerAutoFld</u>	Ausführung auf dem Server, Übertragen der Feldinhalte der verwendeten Felder
<u>SelServerAllFld</u>	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 ersten 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 dazugehörenden Kundendatensätze lesen. Gibt es vergleichsweise wenige Aufträge, müssen dazu wesentlich weniger Datensätze gelesen werden.

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 dazugehörigen 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.

Befehle für dynamische Selektionen

Liste der Befehle und Konstanten zur Bearbeitung von Selektionen

Befehlsgruppen,

Siehe Befehlsliste,

Selektionen

Befehle

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

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:

- '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 {}'
```

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 Abfragenamenname 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

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']
```

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.

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



Abfrage um Verknüpfung erweitern

obj Deskriptor des
 Selektionspuffers
alpha1 Name der übergeordneten
 Abfrage
int2 Nummer der Ziel-Datei
int3 Nummer der Quell-Datei
int4 Nummer der Verknüpfung
alpha5 eigener Abfragenname
 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).

obj -> SelAddResult(int1, int2, int3)
 Selektion Ergebnismenge hinzufügen



obj Deskriptor des
 Selektionspuffers

int1 Nummer der
 Ergebnisdatei

int2 Nummer der
 Ausgangsdatei

int3 Nummer der
 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.

ErrSelResultSet

Die Datei ist bereits als Ergebnismenge definiert oder entspricht der Hauptdatei.

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
Reihenfolge

int3

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-ORDER () 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.

Beispiele:

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

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)

Resultat handle Deskriptor der
neuen Selektion

Verwandte Befehle,

Selection-Objekt,

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 Sortierung
tSel = SelCreate(tblCstCustomer, 0);
tSel->SelDefQuery('', 't
```

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.

obj -> SelDefQuery(alpha1,
alpha2[, alpha3]) : int



Abfrage der Selektion definieren

obj Deskriptor des
 Selektionspuffers

alpha1 Name des Ankers

alpha2 Abfrage

alpha3 Name der
 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

ErrOk

ErrSelUnknownField

ErrSelInvalidField

ErrSelInvalidKey

ErrSelUnknownOrInvalidLink

ErrSelIllegalOperator

ErrSelQueryOverflow

ErrParserEndOfText

ErrParserInvalidChar

ErrParserInvalidConst

Fehler

Kein Fehler

unbekanntes Feld

ungültiges Feld

Vorauswahl mit ungültigem Schlüssel

unbekannte oder ungültige Verknüpfung

unzulässiger Operator

Abfrage zu lang

Abfrage unvollständig

unerlaubtes Zeichen


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.

<u>ErrSource</u>	Selektionsbedingung
<u>ErrCode</u>	Code der Fehlermeldung
<u>ErrText</u>	Text der Fehlermeldung
<u>ErrPos</u>	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(tblCstD
// Alle Kunden, die in einem bestimmten Datumsbereich mindestens einen Auftrag hattentHdlSel # Se
```

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	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.

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 hattentHdlSel # SelCreate(t
```


LinkMax(alpha1,alpha2) : var



Maximalwert unter den verknüpften Sätzen

alpha1 Name der Abfrage

alpha2 Name des Feldes

Resultat var Größter Wert in diesem Feld

Siehe [SelDefQuery\(\)](#), [LinkMin\(\)](#),
[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.

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 angegebene 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.



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.

Beispiel

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

Mögliche Laufzeitfehler:

<u>_ErrHdlInvalid</u>	Der in (obj) übergebene Deskriptor ist ungültig oder nicht vom korrekten Typ.
<u>_ErrValueInvalid</u>	In (alpha1) wurde ein ungültiger oder leerer Name angegeben.
<u>_ErrStringOverflow</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 32 /
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

Siehe Verwandte Befehle, SelOpen(),
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 # tHdlSel->SelClear();
```

Mögliche Laufzeitfehler:

_ErrHdlInvalid Selektionspuffer-Deskriptor (obj) ungültig

obj -> SelClose()



Selektionspuffer entfernen

obj Selektionspuffer-Deskriptor

Siehe Verwandte Befehle,

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

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

Siehe Verwandte Befehle, SelCopy(),
 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(tblCstCustomer, 'SEL_PLZ', 'TMP_' + UserInfo(_UserCurrent));
```

obj -> SelIgnore(int1, logic1) 
 Selektionsvorauswahl ignorieren
 obj Selektionspuffer-Deskriptor
 int1 Schlüsselnummer
 logic2 Ignorieren
 aktivieren/deaktivieren

Siehe Verwandte Befehle,
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

int1 SelFile Selektionsdatei
 SelSort Selektionssortierung
 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 dem 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



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

Alphanumerische Selektionsinformationen ermitteln/setzen

obj Selektionspuffer-Deskriptor

Informationstyp

SelName Selektionsname
ermitteln

SelUser Letzten Benutzer
ermitteln

int1 SelMin Minimalwert ermitteln

SelMax Maximalwert ermitteln

SelRemarks Selektionsbemerkung
ermitteln/setzen

alpha2 Neue Selektionsinformation
(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 Wertmenge 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 Wertmenge 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 Datumsinformationen 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 anlegtHdlSel # SelOpen();tHdlSel->SelRead(tblCstCustomer, _SelLock, 'CstSel
```

Mögliche Laufzeitfehler:

<u>_ErrOutOfMemory</u>	Der Selektionspuffer konnte wegen einem Speicherproblem nicht angelegt werden.
<u>_ErrHdlInvalid</u>	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
gelesen

Resultat int



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

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



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 (beispielweise 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



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
gesperrt

Resultat int rDeadlock Verklemmung ist
aufgetreten

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

Siehe Verwandte Befehle, SelRead(),
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

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



obj Selektionspuffer-Deskriptor

Optionen

<u>SelDisplay</u>	Statusanzeige aktivieren
<u>SelDisplayDelayed</u>	Statusanzeige verzögert aktivieren
<u>SelBreak</u>	Abbruch möglich
<u>SelWait</u>	Statusanzeige nicht automatisch schließen
<u>SelServer</u>	Selektion auf dem Server ausführen
<u>SelServerAllFld</u>	Selektion auf dem Server ausführen und die gefüllten Feldpuffer der gesamten Datenstruktur übertragen
<u>SelServerAutoFld</u>	Selektion auf dem Server ausführen und die in den Selektionselementen verwendeten Feldpuffer übertragen
<u>SelBase</u>	Nachselektion
<u>SelUnion</u>	Vereinigungsmenge bilden
<u>SelInter</u>	Schnittmenge bilden
<u>SelMinus</u>	Restmenge bilden

int1

alpha2 Name der 2. Selektion (optional)

int3 Dateinummer der zweiten Selektion,
sofern abweichend (optional)

int4 Satzlimit (optional)

Resultat int Durchführungsergebnis



<u>rOK</u>	Durchführung erfolgreich
<u>rLocked</u>	2. Selektion (alpha2) gesperrt
<u>rNoKey</u>	2. Selektion (alpha2) kann nicht verwendet werden

<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

Siehe Verwandte Befehle, SelOpen(),
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

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**

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

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->
```

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,

Siehe SelInfo(), SelInfoAlpha(),

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

_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.

SelBreak
Abbruch möglich
Wert 32 /
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.

_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.

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 16 /
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.

_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 1 /
0x00000001

Verwandte
Siehe Befehle,
SelRead(),
SelLast

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

_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 Sortierungsfelder 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 Sortierungsfelder 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 Sortierungsfelder 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 2 /
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.

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.

SelName

Selektionsname ermitteln

Wert 1

Verwandte

Siehe Befehle,

SelInfoAlpha()

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

_SelNext
Nächste Selektion lesen
Wert 4 /
0x00000004

Verwandte
Siehe Befehle,
SelRead(),
_SelPrev

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

SelPrev
Vorherige Selektion lesen
Wert 3 /
0x00000003

Verwandte
Siehe Befehle,
SelRead(),
SelNext

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

SelRemarks
Selektionsbemerkung ermitteln/setzen
Wert 6

Verwandte

Siehe Befehle,

SelInfoAlpha()

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

_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 128 /
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().

SelServerAllFld

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



Wert 512 /
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 256 /
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 48 /
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.

_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 32 /
0x00000020

Verwandte
Siehe Befehle,
SelRead(),
SelStore()

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

SelUnion
Vereinigungsmenge bilden
Wert 2

Verwandte
Siehe Befehle,
SelRun()

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

_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.

SelWait
Statusanzeige nicht automatisch schließen
Wert 64 /
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](#)
- [TextDbas?](#)
- [TextEncrypted](#)
- [TextExtern](#)
- [TextFirst](#)
- [TextGroup](#)
- [TextLast](#)
- [TextLineDelete](#)
- [TextLineInsert](#)
- [TextLines](#)
- [TextLock](#)
- [TextModified](#)
- [TextName](#)
- [TextNext](#)
- [TextNoContents](#)
- [TextNoLineFeed](#)
- [TextOEM](#)
- [TextPrev](#)
- [TextPrivate](#)
- [TextProc](#)
- [TextRemarks](#)

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

obj ->



TextClear()

Textpuffer leeren

obj Textpuffer-Deskriptor


Siehe Verwandte Befehle,

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

Siehe Verwandte Befehle,

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

Siehe Verwandte Befehle, TextCreate(),
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
 TextDb? Datenbankbereich
 Anlegeresultat
 rOk Anlegen
 erfolgreich
Resultat int rExists Text bereits 
 vorhanden
 rDeadlock Verklemmung
 aufgetreten
Siehe Verwandte Befehle, TextCopy(),
 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.

- TextDb?

Der Text wird in einer anderen Datenbank angelegt. 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) zu lang

TextDelete(alpha1, int2) 

: int

Text/Prozedur löschen

alpha1 Text-/Prozedurname

Optionen

int2 0 Internen Text löschen


TextProc Prozedur löschen

TextDb? Datenbankbereich

Löschresultat

rOk Löschen
erfolgreich

rLocked Text (alpha1)
gesperrt

Resultat int rNoRec Text (alpha1) nicht vorhanden 

rNoRights Benutzerrechte
nicht
ausreichend

rDeadlock Verklemmung
aufgetreten

Siehe Verwandte Befehle, TextCreate(),
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.

- TextDb?

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 TextDb2 bis TextDb8 angegeben.

Mögliche Laufzeitfehler:

ErrStringOverflow Textname (alpha1) zu lang



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

Numerische Text-/Prozedurinformationen ermitteln/setzen

obj Textpuffer-Deskriptor

int1 Informationstyp (siehe Text)

int2 Neue Text-/Prozedurinformation
(optional)

Resultat int Aktuelle Text-/Prozedurinformation

Siehe Verwandte Befehle, TextInfoAlpha(),
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

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



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

Alphanumerische Text-/Prozedurinformationen ermitteln/setzen

obj Textpuffer-Deskriptor

Informationstyp

TextName Textname ermitteln

TextUserFirst Benutzer der
Erstellung ermitteln

TextUserLast Benutzer der letzten
Änderung ermitteln

int1 TextUserPrivate Benutzer des privaten
Textes ermitteln

TextGroup Textgruppe
ermitteln/setzen

TextRemarks Textbemerkung
ermitteln/setzen

alpha2 Neue Text-/Prozedurinformation
(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.

- 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

Siehe Verwandte Befehle, TextInfo(),
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 Zeitinformatoren (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

obj -> TextLineRead(int1, int2) : 

alpha

Text-/Prozedurzeile lesen/löschen

obj Textpuffer-Deskriptor

int1 Zeilennummer

Optionen

int2 TextLineDelete Zeile
 löschen

Resultat alpha Zeileninhalt

Verwandte Befehle,

Siehe TextOpen(),

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

Siehe Verwandte Befehle, TextOpen(),
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)
 Leseresultat
 _rOk Lesen
 erfolgreich
 _rLocked Text
 (alpha1)
 gesperrt
 _rNoKey Text
 (alpha1)
 nicht
 vorhanden,
 nächster
 Resultat int Text 
 gelesen
 _rNoRec Kein
 weiterer
 Text
 vorhanden
 _rLastRec Text
 (alpha1)
 nicht
 vorhanden,
 letzter Text
 gelesen

Siehe [Verwandte Befehle](#),
[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, int3) : int

Text/Prozedur umbenennen

alpha1 Text-/Prozedurname

alpha2 Neuer Text-/Prozedurname

Optionen

int3 TextProc Prozedur
umbenennen


TextDb? Datenbankbereich

Löschresultat

rOk Umbenennen
erfolgreich

rLocked Text (alpha1)
gesperrt

rNoRec Text (alpha1)
nicht
vorhanden

Resultat int rExists 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

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

Zeichenfolge in einem Text suchen / ersetzen

obj Textpuffer-Deskriptor

int1 Anfangszeile

int2 Anfangsspalte

 Optionen

TextSearchCI Groß-/Kleinwandlung

int3 TextSearchCount Suchbegriffsfunde

TextSearchToken Begriffsorientierte
 Suche

alpha4 Suchbegriff

alpha5 Ersatzbegriff (optional)

int6 Ersetzungsanzahl (optional)

 Suchresultat

 Zeilennummer

Resultat int Suchbegriffsfunde

 Ersetzungsvorgänge

Verwandte Befehle, TextOpen(),

Siehe 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, 'CONZEPT 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



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->TextSearchRegEx(obj, int1, int2, int3, alpha4, &resultat))
```

Mit der Funktion TextInfo() mit der Option TextSearchClm 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

ErrRegExRuleSyntax

ErrRegExBadEscapeSequence

ErrRegExPropertySyntax

ErrRegExNotSupported

ErrRegExMismatchedParentheses

ErrRegExNumberTooBig

ErrRegExBadInterval

Bedeutung

Syntaxfehler im regulären Ausdruck

Nicht aufgelöste Escape-Sequenz im Ausdruck

Ungültige Unicode-Eigenschaft

Verwendung einer Funktion, die nicht unterstützt wird

Falsch verschachtelte Klammern im regulären Ausdruck

Dezimalzahl zu groß

Fehler im {min,max} Intervall

Kontakt

<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 Klammersausdruck
<u>ErrRegExInvalidRange</u>	In einer Zeichenmenge [x-y] ist x größer als y
<u>ErrRegExStackOverflow</u>	Stapelüberlauf in der Ablaufverfolgung des regulären Ausdrucks

Mögliche Laufzeitfehler:

ErrHdlInvalid Textpuffer-Deskriptor (obj) ungültig

obj -> TextWrite(alpha1,
int2) : int



Text/Prozedur schreiben

obj Textpuffer-Deskriptor


alpha1 Textname

int2 Optionen (siehe Text)

Schreibresultat

rOk Schreiben erfolgreich

rLocked Text (alpha2) gesperrt

Resultat int rNoRights Benutzerberechtigung 
nicht ausreichend

rDeadlock Verklemmung
aufgetreten

Siehe Verwandte Befehle, TextOpen(),
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
- TextDbas?
- TextEncrypted
- TextExtern
- TextFirst
- TextGroup
- TextLast
- TextLineDelete
- TextLineInsert
- TextLines
- TextLock
- TextModified
- TextName
- TextNext
- TextNoContents
- TextNoLineFeed
- TextOEM
- TextPrev
- TextPrivate
- TextProc
- TextRemarks
- TextRightRead
- TextRightWrite
- TextSearchCI
- TextSearchClm
- 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.

_TextAuthRead
Leserecht ermitteln
Wert 9

Verwandte

Siehe Befehle,

TextInfo()

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

_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
Wert -2.147.483.648
/ 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.

_TextDba?

Text in anderer Datenbank ansprechen

Wert 0x01000000 -
0x07000000

Siehe Textbefehle,
DbaConnect()

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 DbaConnect() verbunden werden. Der dabei angegebene Nummernbereich bestimmt mit welcher Option Texte dieser Datenbank angesprochen werden können:

Nummernbereich 2 : _TextDba2

Nummernbereich 3 : _TextDba3

Nummernbereich 4 : _TextDba4

Nummernbereich 5 : _TextDba5

Nummernbereich 6 : _TextDba6

Nummernbereich 7 : _TextDba7

Nummernbereich 8 : _TextDba8

_TextEncrypted
Verschlüsselungsmarkierung ermitteln/setzen
Wert 6

Verwandte

Siehe Befehle,

TextInfo()

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

_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 1 /
0x00000001

Verwandte
Siehe Befehle,
TextRead(),
_TextLast

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

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 2 /
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.

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.

_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 4 /
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 3 /
0x00000003

Verwandte
Siehe Befehle,
TextRead(),
_TextNext

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

_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.

TextRemarks
Textbemerkung ermitteln/setzen
Wert 6

Verwandte

Siehe Befehle,

TextInfoAlpha()

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

_TextRightRead
Leseberechtigung ermitteln
Wert 3

Verwandte

Siehe Befehle,

TextInfo()

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

_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.

_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.

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.

_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.

_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.

_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.

_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.

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

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

int

Objekt der Benutzerverwaltung erzeugen

obj Deskriptor des Eltern-Objekts oder 0

Typ des Objekts

_UrmTypeUser Benutzer

_UrmTypeUserGroup Benutzergruppe

_UrmTypeElmGroup Elementgruppe

int1 _UrmTypeProperty Benutzerdefinierte
Eigenschaft

_UrmTypeMember Mitglied oder
Benutzergruppe

_UrmTypeElm... Element einer
Elementgruppe

alpha2 Name des Objekts

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

_TypeAlpha Alphanumerisch

_TypeBigInt Ganzzahlig (64 Bit)

_TypeDate Datum

int3 _TypeDecimal Dezimal

_TypeFloat Gleitkomma

_TypeInt Ganzzahlig (32 Bit)

_TypeLogic Logisch

_TypeTime 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

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.

UrmOpen(int1, int2,
alpha3) : handle
Benutzerobjekt öffnen



Typ
int1 UrmTypeUser Benutzer
 UrmTypeUserGroup Benutzergruppe
 UrmTypeElmGroup Elementgruppe
Sperrmodus
0 nicht sperren
int2 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 Datensatzsperren 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 ...}...
```

Mögliche Laufzeitfehler:

_ErrValueInvalid In (int1) wurde ein ungültiger Typ übergeben.

_ErrStringOverflow Der Objektname in (alpha3) ist zu lang.

obj -> UrmClose()



Benutzerobjekt schließen

Deskriptor des

obj Objekts der
Benutzerpflege

Verwandte

Befehle,

Siehe 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
```



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**

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 Benutzergruppe
```

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.


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
 verboten

bigint3 Berechtigungsflags

Resultat logic Rechte vorhanden 

Siehe Verwandte Befehle,
 Benutzerpflege,
 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

_UrmPermMemberInsert Berechtigung Benutzer zu Benutzergruppe hinzufügen
_UrmPermMemberDelete 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,

Siehe Benutzerpflege,

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

_UrmPermMemberInsert Berechtigung Benutzer zu Benutzergruppe hinzufügen
_UrmPermMemberDelete 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)\
```

Mögliche Laufzeitfehler:

_ErrHdlInvalid 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.

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

<u>ErrStringOverflow</u>	Es wurde ein zu langer (mehr als 80 Zeichen) Elementname angegeben.
<u>ErrValueInvalid</u>	Es wurde eine unbekannte Berechtigung (bigint2) übergeben.

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

_UrmPermElmGroupRead Berechtigung Benutzer zu Benutzergruppe hinzufügen
_UrmPermElmGroupInsert 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

int2 UrmAllow Recht
 erlauben

UrmDeny Recht
 verbieten

bigint3 Berechtigungsflags

Resultat int Fehlerwert

[Verwandte Befehle,](#)

Siehe [Benutzerpflege,](#)

[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

_UrmPermMemberInsert Berechtigung Benutzer zu Benutzergruppe hinzufügen
_UrmPermMemberDelete 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 Neuansmeldung 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.



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.



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

obj Deskriptor der
 Benutzergruppe

alpha1 Name der
 Elementgruppe

int2 Neuer
 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.

ErrOk Kein Fehler aufgetreten

ErrRights Berechtigung nicht ausreichend

ErrUrmObjectNotFound Das angegebene Objekt (obj) wurde nicht gefunden

ErrUnchangeable Berechtigungslevel kann nicht geändert werden

Beispiel

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

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

ErrStringOverflow Es wurde ein zu langer Objektname angegeben.

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.

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 Eigenschaft
```

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.


obj -> UrmPropType(alpha1) : 

int

Typ einer Eigenschaft ermitteln

obj Deskriptor eines Objekts
 der Benutzerverwaltung

alpha1 Name oder Konstante der
 Eigenschaft

Resultat int Typ der Eigenschaft 

Siehe Verwandte Befehle,
 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)



	<u>UrmTypeUser</u>	Benutzer lesen
	<u>UrmTypeUserGroup</u>	Benutzergruppe lesen
	<u>UrmTypeElmGroup</u>	Elementgruppe lesen
int1	<u>UrmTypeSysProperty</u>	Vordefinierte Eigenschaft lesen
	<u>UrmTypeProperty</u>	Benutzerdefinierte Eigenschaft lesen
	<u>UrmTypeMember</u>	Mitglied oder Benutzergruppe lesen
	<u>UrmTypePerm</u>	Recht lesen
	<u>UrmTypeElm...</u>	Element lesen
	Lesemodus	
	0	Das angegebene Objekt (alpha4) lesen
int2	<u>UrmFirst</u>	erstes Objekt lesen
	<u>UrmPrev</u>	vorheriges Objekt lesen
	<u>UrmNext</u>	nachfolgendes Objekt lesen
	<u>UrmLast</u>	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

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);loc
```

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

_PwModify

Passwort ändern

Wert 2 / 0x02

Verwandte

Siehe Befehle,

UserPassword()

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

_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.

_UrmFirst
Erstes Objekt lesen
Wert 1 /
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.

_UrmLast
Lettes Objekt lesen
Wert 2 /
0x00000002

Verwandte

Siehe Befehle,

UrmRead()

Option beim Befehl UrmRead() - das letzte Objekt wird gelesen.

_UrmLock
Benutzerobjekt sperren

Wert 8 /
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.

_UrmNext
Nächstes Objekt lesen
Wert 4 /
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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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(),
UrmPermSet()

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 -2.147.483.648 /
0x80000000

Verwandte
Befehle,
Konvertierung
des alten
Siehe Benutzersystems,
UrmPermGet(),
UrmPermGetRaw(),
UrmPermSet()

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.

_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 16 /
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 64 /
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.

_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.

_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.

_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 32 /
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 | _
```

_UrmPrev
Vorhergehendes Objekt lesen
Wert 3 /
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.

_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.

_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.

_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.

_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.

_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.

_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.

_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

Abmelden
 erfolgreich

ErrDbidUserInvalid

Benutzer nicht
 eingeloggt oder
 Benutzer-ID (int1)
 passt nicht zu
 Benutzernummer
 (int2)

Resultat int

ErrDbidUserSelf

Eigene
 Benutzer-ID in
 (int1) angegeben,
 Abmelden nicht
 möglich

ErrDbidAreaPassword

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.

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



UserDelete(alpha1) : int

Datenbankbenutzer löschen

alpha1 Name des Benutzers

Resultat des Löschvorgangs

__ErrOk

Löschen des
Benutzers
erfolgreich

__ErrLocked

Der Benutzer
(alpha1) ist
gesperrt

Resultat int __ErrUrmObjectNotFound Der Benutzer
(alpha1) ist
nicht

vorhanden

__ErrRights

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
 ermitteln
 int1
 UserLocked Sperrenden
 Benutzer
 ermitteln
 Resultat int Benutzer-ID
 Verwandte Befehle,
 Siehe UserInfo(),
 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.

UserInfo(int1[, int2[, int3]]) : 

alpha

Benutzerinformationen ermitteln

int1 Informationstyp (siehe Text)

int2 Benutzer-ID (optional)

int3 Verbundene Datenbank (optional)

Resultat alpha Benutzerinformation

Verwandte Befehle,

Siehe UserClear(),

UserPassword(),

DbaInfo(), 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 DbaConnect()) die angegebene Information ermittelt werden soll. Es können die Konstanten _Dba2 bis _Dba8 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

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

int1

aktuellen

Benutzers

_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
 int1 aktuellen
 Benutzers
 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

UserPassword(alpha1, alpha2, alpha3[, int4) : int



Benutzerpasswort ändern

alpha1 Benutzername

alpha2 altes Benutzerpasswort

alpha3 neues Benutzerpasswort

Optionen (optional)

int4 PwdModify Passwort ändern

PwdVerify Altes Passwort überprüfen

Änderungsergebnis

ErrOK

Änderung erfolgreich

ErrUrmParentNotFound

Benutzer (alpha1) nicht vorhanden

ErrRights

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.



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

_UserAddress
Netzwerkadresse ermitteln
Wert 3

Verwandte
Siehe Befehle,
UserInfo()

Option bei UserInfo() durch die die Netzwerkadresse (IP-Adresse) des Clients ermittelt werden kann.

_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.

_UserJobID

JobID ermitteln

Wert 18

Verwandte

Siehe Befehle,

UserInfo()

Option bei UserInfo() durch die die JobID eines Benutzers ermittelt werden kann.

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.

_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.

_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.

_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.

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.

_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.

_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.

_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.

_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 ('').

_UserNextID

Nächste Benutzer-ID ermitteln

Wert 1

Verwandte

Siehe Befehle,

UserInfo()

Option bei UserInfo() durch die die Nächste Benutzer-ID ermittelt werden kann.

_UserNumber
Benutzernummer ermitteln
Wert 7

Verwandte
Siehe Befehle,
UserInfo()

Option bei UserInfo() durch die die Nummer des Benutzers ermittelt werden kann.

_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.

_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.

_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.

_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).

_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

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

_ErrGeneric

Datenbank ist
mit dem
Advanced-Client
geöffnet

_ErrOemDbalock

Datenbank
gesperrt


_ErrOemOpenFailed

Definitionsdatei
kann nicht
geöffnet werden

_ErrOemInvalidFormat

Definitionsdatei
hat falsches

Resultat int

Format oder 
wurde mit
anderer
CONCEPT
16-Version
erstellt

_ErrOemOutOfSpace

Nicht genügend
Speicher zum
Laden verfügbar

_ErrOemOpenDesigner

Prozedur wurde
aus Designer
gestartet

_ErrOemOpenFrame

Noch
mindestens ein
Dialog geöffnet

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 CONCEPT 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:\C16\Definition', '', var t.aErr) != _ErrOk){ WinDialogBox(0, 'Fehler', t.aErr, _
```

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

_ErrOemPassword OEM-Kit-Passwort
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

_OEMWait
Statusanzeige nicht automatisch schließen
Wert 1 /
0x00000001

Verwandte

Siehe Befehle,

OEMSave()

Option bei OEMSave() durch die ein automatisches Schließen der Statusanzeige nach dem Exportieren verhindert werden kann.

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



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:

_ErrHdllInvalid Deskriptor des Validierungsverzeichnisses bzw. Validierungselementes (obj) ist ungültig.



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:

ErrOk Löschen erfolgreich
ErrVldNameInvalid Elementname (alpha1) ist ungültig
ErrVldNoFile Validierungselement (alpha1) existiert nicht
ErrVldLocked Validierungselement (alpha1) ist gesperrt
rDeadlock Verklemmung aufgetreten

Beispiel

```
// Objekt 'Frm_Main' im Verzeichnis tVldDir löschtVldDir->VldDelete('Frm_Main');
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor des Validierungsverzeichnisses (obj) ist ungültig.



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
 Validierungselement
 lesen
 int1 _VldNext Validierungselement
 nach
 Referenzeintrag
 lesen
 _VldPrev Validierungselement
 vor Referenzeintrag
 lesen

alpha2 Referenzeintrag (optional)

Resultat alpha Eintragsname

Siehe Verwandte Befehle,
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.



obj -> VldOpen(alpha1[, int2]) : handle
Validierungselement öffnen bzw. erstellen
obj Validierungsverzeichnis
alpha1 Elementname
Optionen (optional)
int2 _VldLock Element für andere Benutzer 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
_rDeadlock Verklemmung 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');// Objekt
```

Mögliche Laufzeitfehler:

_ErrHdlInvalid Deskriptor des Validierungsverzeichnisses (obj) ist ungültig.

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 1 /
0x00000001

Verwandte
Siehe Befehle,
VldDirRead()

Option bei VldDirRead() durch die das erste Validierungselement in einem Validierungsverzeichnis gelesen werden kann.

VldLast
Letztes Validierungselement lesen
Wert 2 /
0x00000002

Verwandte
Siehe Befehle,
VldDirRead()

Option bei VldDirRead() durch die das letzte Validierungselement in einem Validierungsverzeichnis gelesen werden kann.

VldLock
Validierungselement für andere Benutzer sperren
Wert 8 /
0x00000008

Verwandte
Siehe Befehle,
VldOpen()

Option bei VldOpen() durch die ein Validierungselement beim Öffnen/Anlegen für andere Benutzer gesperrt werden kann.

VldNext
Element nach Referenzeintrag lesen
Wert 4 /
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 3 /
0x00000003

Verwandte

Siehe Befehle,

VldDirRead()

Option bei VldDirRead() durch die das Validierungselement vor dem Referenzeintrag in einem Validierungsverzeichnis gelesen werden kann.

_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.

_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.

_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

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



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-Desriptor (obj) ungültig



obj -> FsiDate(int1[, date2]) : date

Dateiinformatioren 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

Siehe Verwandte Befehle, FsiOpen(),
 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

FsiDelete(alpha1[, int2]) : 

int

Datei löschen

alpha1 Dateiname/-pfad

Optionen (optional)

_FsiNameC16 Dateiname/-pfad (alpha1) wird
im CONZEPT 16-Zeichensatz
erwartet (Standard)

int2

_FsiNameUtf8 Dateiname/-pfad (alpha1) wird
im UTF-8-Zeichensatz
erwartet

Löschergebnis


_ErrOk

Löschen
erfolgreich

_ErrFsiNoFile

Dateiname/-pfad
(alpha1) nicht
vorhanden

_ErrFsiAccessDenied

Berechtigungen
nicht
ausreichend 

Resultat int

_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


Siehe Verwandte Befehle,

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 VerzeichnisDirHdl # FsiDirOpen('*.dat', _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

Siehe Verwandte Befehle,
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



FsiDiskInfo(alpha1, int2) : int
 Datenträgerinformationen ermitteln
 alpha1 Laufwerksbuchstabe

Dateiattribute

	<u>FsiDiskFree</u>	Freie Kapazität in KB ermitteln
	<u>FsiDiskFreeMB</u>	Freie Kapazität in MB ermitteln
	<u>FsiDiskTotal</u>	Gesamte Kapazität in KB ermitteln
int2	<u>FsiDiskTotalMB</u>	Gesamte Kapazität in MB ermitteln
	<u>FsiDiskAvailMB</u>	Verfügbare Kapazität in MB ermitteln
	<u>FsiDiskReady</u>	Datenträgerbereitschaft ermitteln
	<u>FsiDiskExists</u>	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

int2 ComprFmtDeflate DEFLATE-Format

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,

Siehe FsiFileUncompress(),
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.txt', 'Test.txt.gz', ComprFmtGzip)
```

Fehlerwerte

Zusätzlich zu den Fehlerwerten für externe Dateioperationen (ErrFsi...) können folgende Fehlerwerte von der Funktion zurückgegeben werden:

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.



FsiFileInfo(alpha1, int2) : alpha
Informationen zu einer externen Datei ermitteln

alpha1 Pfad und Dateiname der
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 kollisionssicher 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 verändert
```

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 Datei anhängen

FsiFileCRC32 CRC32-Prüfsumme an die Datei anhängen

int3 FsiFileRMD160 RIPEMD-160-Hash-Wert an die Datei anhängen

FsiFileSHA1 SHA-1-Hash-Wert an die Datei anhängen

FsiFileSHA256 SHA-256-Hash-Wert an die Datei anhängen

FsiFileSHA384 SHA-384-Hash-Wert an die Datei anhängen

FsiFileSHA512 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:

- 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 Dateinhalt 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.



Die Verfahren FsiFileCRC32, FsiFileMD5, FsiFileRMD160 und FsiFileSHA1 sind nicht kollisionssicher 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

FsiFileUncompress(alpha1[,
int2[, int3[, alpha4]]) : int
Externe Dateien dekomprimieren



alpha1 Name der
Quelldatei
int2 Quellposition
(optional)
int3 Quelllänge
(optional)
alpha4 Name der
Zieldatei
(optional)
Resultat int Fehlerwert 

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



obj -> FsiMark([int1]) : byte
 Dateitrennzeichen ermitteln/setzen
 obj Datei-Deskriptor
 int1 Neues Dateitrennzeichen
 (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

obj -> FsiMonitorAdd(alpha1[, 
int2[, alpha3[, alpha4]]])

Verzeichnis überwachen

obj Deskriptor der Verzeichnisüberwachung

alpha1 Name des zu überwachenden
Verzeichnisses

Option (optional)

FsiMonFlagsSubDirs Überwachung
der
untergeordneten
Verzeichnisse

int2

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" überwachen FsiMonitor->FsiMonitorAdd(_Sys->spPathMyDc
```

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.



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 überwachtetes Verzeichnis

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 

Verwandte Befehle, EvtFsiMonitor,

Siehe 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
Verwandte Befehle,
 Siehe 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

- 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

- _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
int1 CONZEPT 16-Zeichensatz
ermittelt (Standard)

FsiNameUtf8 Aktuelles Verzeichnis wird
im UTF-8-Zeichensatz
ermittelt

Resultat alpha Aktuelles Verzeichnis

Siehe Verwandte Befehle, FsiPathChange(),
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.

FsiPathChange(alpha1[,
 int2]) : int
 Verzeichnis wechseln
 alpha1 Neues Verzeichnis
 Optionen (optional)
 FsiNameC16 Neues Verzeichnis
 (alpha1) wird im
 CONZEPT 16-Zeichensatz
 erwartet (Standard)
 int2
 FsiNameUtf8 Neues Verzeichnis
 (alpha1) wird im
 UTF-8-Zeichensatz
 erwartet
 Resultat int Fehlerwert



Siehe [Verwandte Befehle](#), [FsiPath\(\)](#),
[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.

FsiPathCreate(alpha1[,  int2]) : int

Verzeichnis erstellen

alpha1 Verzeichnisname

Optionen (optional)

_FsiNameC16 Verzeichnisname (alpha1)
wird im
CONZEPT 16-Zeichensatz
erwartet (Standard)

int2
_FsiNameUtf8 Verzeichnisname (alpha1)
wird im
UTF-8-Zeichensatz
erwartet

Resultat int Fehlerwert 

Siehe Verwandte Befehle, FsiPathChange(),
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.

FsiPathDelete(alpha1[,
 int2]) : int
 Verzeichnis löschen
 alpha1 Verzeichnisname
 Optionen (optional)
 FsiNameC16 Verzeichnisname (alpha1)
 wird im
 CONZEPT 16-Zeichensatz
 erwartet (Standard)
 int2
 FsiNameUtf8 Verzeichnisname (alpha1)
 wird im
 UTF-8-Zeichensatz
 erwartet

Resultat int Fehlerwert 

Siehe Verwandte Befehle, FsiPathCreate(),
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)**

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: tAlpha->FsiRead(tAlpha, 20); // Einlesen eines ASCII-Werts
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Deskriptor (obj) ist ungültig.

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

Resultat int Anzahl der gelesenen Bytes 
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

alpha1 Alter
 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') = _Err0k){ ...}
```