

Kontakt

obj -> ComEvtProcessGet(int1) : logic



Ermitteln, ob ein COM-Ereignis aktiviert oder deaktiviert ist

obj Objekt

int1 Konstante des Ereignisses

Resultat logic Aktivierung des Ereignisses

Verwandte Befehle,

Siehe

ComEvtProcessSet(), CtxDocEdit

Mit diesem Befehl kann ermittelt werden, ob das in (int1) angegebene COM-Ereignis bei dem Objekt (obj) aktiviert (Resultat = true) oder deaktiviert (Resultat = false) ist.

Eine Liste der Ereignis-Konstanten ist in der Prozedur CtxDocEdit.Define.prc in dem CodeLibrary-Beispiel "CtxDocEdit" enthalten. Diese Prozedur kann auch in der eigenen Applikation includiert werden. Die Liste der Ereignisse und deren Beschreibung finden Sie zusätzlich auf der Hersteller-Seite des Moduls. Die dort genannten Ereignisse können dann mit dem Präfix txEvt verwendet werden.

Beispiel

```
if (!$_ctxDocEdit->ComEvtProcessGet(txEvtPosChange)) { // Bei der Änderung der Cursorposition wird
```

Kontakt

obj -> ComEvtProcessSet(int1, logic2) : logic  Aktivieren oder

Deaktivieren eines COM-Ereignisses

obj Objekt

int1 Konstante des Ereignisses

Ereignis aktivieren (true) oder

logic2 deaktivieren (false)

Resultat logic vorhergehender Zustand

Verwandte Befehle,

Siehe ComEvtProcessGet(),

CtxDocEdit

Mit diesem Befehl kann die Verarbeitung eines COM-Ereignisses aktiviert oder unterdrückt werden.

In (int1) wird die Ereignis-Nummer übergeben. Wird in (logic2) false übergeben, wird dieses

Ereignis bei dem Objekt (obj) nicht mehr ausgelöst.

Eine Liste der Ereignis-Konstanten ist in der Prozedur CtxDocEdit.Define.prc in dem CodeLibrary-Beispiel "CtxDocEdit" enthalten. Diese Prozedur kann auch in der eigenen Applikation inkludiert werden. Die Liste der Ereignisse und deren Beschreibung finden Sie zusätzlich auf der Hersteller-Seite des Moduls. Die dort genannten Ereignisse können dann mit dem Präfix txEvt verwendet werden.

Wird ein aktiviertes COM-Ereignis ausgelöst, wird das Ereignis EvtCtxEvent des Objektes (obj) aufgerufen. In dem Parameter (aEventID) wird die Nummer des COM-Ereignisses übergeben.

Der Rückgabewert enthält den vorhergehenden Aktivierungszustand. Ob ein COM-Ereignis aktiviert oder deaktiviert ist, kann mit dem Befehl ComEvtProcessGet() ermittelt werden.

Beispiele:

```
// Ereignis bei Änderung der Cursor-Position aktivieren$ctxDocEdit->ComEvtProcessSet(txEvtPosChan
```

Kontakt



WinEvtProcessGet(int1) : logic

Ermitteln, ob ein Ereignis aktiviert oder deaktiviert ist int1

Konstante des Ereignisses

Resultat logic Aktivierung des Ereignisses

Siehe

Verwandte Befehle,
WinEvtProcessSet(),

WinEvtProcNameGet(),
WinEvtProcNameSet()

Mit diesem Befehl kann ermittelt werden, ob das in (int1) angegebene Ereignis aktiviert (Resultat = true) oder deaktiviert (Resultat = false) ist.

Folgende Konstanten können in (int1) angegeben werden:

- WinEvtAdviseDDE
- WinEvtAttachState
- WinEvtChanged
- WinEvtChangedActive
- WinEvtChangedChild
- WinEvtChangedDesign
- WinEvtClicked
- WinEvtClose
- WinEvtCreated
- WinEvtCroNavigate
- WinEvtCtxEvent
- WinEvtDbFldUpdate
- WinEvtDragInit
- WinEvtDragTerm
- WinEvtDrop
- WinEvtDropEnter
- WinEvtDropLeave
- WinEvtDropOver
- WinEvtEndSession
- WinEvtFocusCancel
- WinEvtFocusInit
- WinEvtFocusTerm
- WinEvtFsiMonitor
- WinEvtHelpTip
- WinEvtHyphenate
- WinEvtInit
- WinEvtIvlDropItem
- WinEvtJob
- WinEvtKeyItem
- WinEvtLstDataInit
- WinEvtLstEditActivate
- WinEvtLstEditCommit
- WinEvtLstEditEndGroup
- WinEvtLstEditEndItem
- WinEvtLstEditFinished
- WinEvtLstEditStart
- WinEvtLstEditStartGroup

Kontakt

- [WinEvtLstEditStartItem](#)
- [WinEvtLstGroupArrange](#)
- [WinEvtLstGroupInit](#)
- [WinEvtLstRecControl](#)
- [WinEvtLstSelect](#)
- [WinEvtLstSelectRange](#)
- [WinEvtLstViewInit](#)
- [WinEvtMdiActivate](#)
- [WinEvtMenuCommand](#)
- [WinEvtMenuContext](#)
- [WinEvtMenuInitPopup](#)
- [WinEvtMenuPopup](#)
- [WinEvtMouse](#)
- [WinEvtMenuItem](#)
- [WinEvtMouseMove](#)
- [WinEvtNodeExpand](#)
- [WinEvtNodeSearch](#)
- [WinEvtNodeSelect](#)
- [WinEvtPageSelect](#)
- [WinEvtPosChanged](#)
- [WinEvtReadOnlyChanged](#)
- [WinEvtSocket](#)
- [WinEvtSystem](#)
- [WinEvtTapi](#)
- [WinEvtTerm](#)
- [WinEvtTimer](#)
- [WinEvtUser](#)

Beispiel

```
if (!WinEvtProcessGet(_WinEvtFocusTerm)){ // EvtFocusTerm ist deaktiviert ...}
```

Kontakt

WinEvtProcessSet(int1, logic2) : logic  Aktivieren oder

Deaktivieren eines Ereignisses

int1 Konstante des Ereignisses
Ereignis aktivieren (true) oder
logic2 deaktivieren (false)

Resultat logic vorhergehender Zustand

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

Siehe [WinEvtProcNameGet\(\)](#),
[WinEvtProcNameSet\(\)](#)

Mit diesem Befehl kann die Verarbeitung eines Ereignisses oder aller Ereignisse unterdrückt werden. In (int1) wird das Ereignis übergeben. Wird in (logic2) false übergeben, wird dieses Ereignis bei allen Objekten nicht mehr ausgelöst.

Das Ereignis wird erst dann wieder ausgelöst, wenn es mit (logic2) = true wieder aktiviert wird.

Folgende Konstanten können in (int1) angegeben werden:

- [WinEvtAdviseDDE](#)
- [WinEvtAttachState](#)
- [WinEvtChanged](#)
- [WinEvtChangedActive](#)
- [WinEvtChangedChild](#)
- [WinEvtChangedDesign](#)
- [WinEvtClicked](#)
- [WinEvtClose](#)
- [WinEvtCreated](#)
- [WinEvtCroNavigate](#)
- [WinEvtCtxEvent](#)
- [WinEvtDbFldUpdate](#)
- [WinEvtDragInit](#)
- [WinEvtDragTerm](#)
- [WinEvtDrop](#)
- [WinEvtDropEnter](#)
- [WinEvtDropLeave](#)
- [WinEvtDropOver](#)
- [WinEvtEndSession](#)
- [WinEvtFocusCancel](#)
- [WinEvtFocusInit](#)
- [WinEvtFocusTerm](#)
- [WinEvtFsiMonitor](#)
- [WinEvtHelpTip](#)
- [WinEvtHyphenate](#)
- [WinEvtInit](#)
- [WinEvtIvlDropItem](#)
- [WinEvtJob](#)
- [WinEvtKeyItem](#)
- [WinEvtLstDataInit](#)
- [WinEvtLstEditActivate](#)

Kontakt

- [WinEvtLstEditCommit](#)
- [WinEvtLstEditEndGroup](#)
- [WinEvtLstEditEndItem](#)
- [WinEvtLstEditFinished](#)
- [WinEvtLstEditStart](#)
- [WinEvtLstEditStartGroup](#)
- [WinEvtLstEditStartItem](#)
- [WinEvtLstGroupArrange](#)
- [WinEvtLstGroupInit](#)
- [WinEvtLstRecControl](#)
- [WinEvtLstSelect](#)
- [WinEvtLstSelectRange](#)
- [WinEvtLstViewInit](#)
- [WinEvtMdiActivate](#)
- [WinEvtMenuCommand](#)
- [WinEvtMenuContext](#)
- [WinEvtMenuInitPopup](#)
- [WinEvtMenuPopup](#)
- [WinEvtMouse](#)
- [WinEvtMenuItem](#)
- [WinEvtMouseMove](#)
- [WinEvtNodeExpand](#)
- [WinEvtNodeSearch](#)
- [WinEvtNodeSelect](#)
- [WinEvtPageSelect](#)
- [WinEvtPosChanged](#)
- [WinEvtReadOnlyChanged](#)
- [WinEvtSocket](#)
- [WinEvtSystem](#)
- [WinEvtTapi](#)
- [WinEvtTerm](#)
- [WinEvtTimer](#)
- [WinEvtUser](#)

Sollen alle Ereignisse aktiviert oder deaktiviert werden, kann die Konstante [WinEvtAll](#) verwendet werden.

Der Rückgabewert enthält den vorhergehenden Aktivierungszustand. Ob ein Ereignis aktiviert oder deaktiviert ist, kann mit dem Befehl [WinEvtProcessGet\(\)](#) ermittelt werden.



Ereignisse, die nur dann aufgerufen werden, wenn ein anderes Ereignis einen bestimmten Wert zurück gibt (zum Beispiel [EvtFocusCancel](#)), werden durch die Deaktivierung ebenfalls nicht mehr ausgeführt.

Beispiele:

```
// Keine Ereignisse mehr auslösenWinEvtProcessSet(_WinEvtAll, false); // Alle Ereignisse wieder ak
```

Kontakt

obj -> WinEvtProcNameGet(int1) : alpha | Ermittelt den

Funktionsnamen eines Ereignisses

obj Fenster-Objekt

int1 Konstante des Ereignisses

Resultat **alpha** Prozedur und Funktionsname

Siehe

Verwandte Befehle,
WinEvtProcessGet(),

WinEvtProcessSet(),
WinEvtProcNameSet()

Der Befehl liefert den Namen der Ereignissprozedur zurück, die beim Objekt (obj) beim Ereignis (int1) hinterlegt ist.

Folgende Konstanten können in (int1) angegeben werden:

- WinEvtAdviseDDE
- WinEvtAttachState
- WinEvtChanged
- WinEvtChangedActive
- WinEvtChangedChild
- WinEvtChangedDesign
- WinEvtClicked
- WinEvtClose
- WinEvtCreated
- WinEvtCrxNavigate
- WinEvtCtxEvent
- WinEvtDbFldUpdate
- WinEvtDragInit
- WinEvtDragTerm
- WinEvtDrop
- WinEvtDropEnter
- WinEvtDropLeave
- WinEvtDropOver
- WinEvtEndSession
- WinEvtFocusCancel
- WinEvtFocusInit
- WinEvtFocusTerm
- WinEvtFsiMonitor
- WinEvtHelpTip
- WinEvtHyphenate
- WinEvtInit
- WinEvtIvlDropItem
- WinEvtJob
- WinEvtKeyItem
- WinEvtLstDataInit
- WinEvtLstEditActivate
- WinEvtLstEditCommit
- WinEvtLstEditEndGroup
- WinEvtLstEditEndItem
- WinEvtLstEditFinished
- WinEvtLstEditStart

Kontakt

- [WinEvtLstEditStartGroup](#)
- [WinEvtLstEditStartItem](#)
- [WinEvtLstGroupArrange](#)
- [WinEvtLstGroupInit](#)
- [WinEvtLstRecControl](#)
- [WinEvtLstSelect](#)
- [WinEvtLstSelectRange](#)
- [WinEvtLstViewInit](#)
- [WinEvtMdiActivate](#)
- [WinEvtMenuCommand](#)
- [WinEvtMenuContext](#)
- [WinEvtMenuInitPopup](#)
- [WinEvtMenuPopup](#)
- [WinEvtMouse](#)
- [WinEvtMenuItem](#)
- [WinEvtMouseMove](#)
- [WinEvtNodeExpand](#)
- [WinEvtNodeSearch](#)
- [WinEvtNodeSelect](#)
- [WinEvtPageSelect](#)
- [WinEvtPosChanged](#)
- [WinEvtReadOnlyChanged](#)
- [WinEvtSocket](#)
- [WinEvtTapi](#)
- [WinEvtTerm](#)
- [WinEvtTimer](#)
- [WinEvtUser](#)

Beispiel

```
tHdl->WinEvtProcNameGet(_WinEvtTimer) // -> "Prozedur: EvtTimer"
```

Kontakt

obj -> WinEvtProcNameSet(int1,
alpha2) : logic



Ereignisfunktion eines Objekts setzen

obj Deskriptor des Objekts

int1 ID des Ereignisses

alpha2 Name der Ereignisfunktion

Resultat logic Erfolg der Anweisung

Verwandte Befehle,
WinEvtProcessGet(),

Siehe

WinEvtProcessSet(),
WinEvtProcNameGet()

Mit dem Befehl kann bei dem angegebenen Objekt (obj) eine Ereignisfunktion gesetzt werden.

Die ID des Ereignisses wird in (int1) übergeben. Folgende Konstanten können übergeben werden:

- WinEvtAdviseDDE
- WinEvtAttachState
- WinEvtChanged
- WinEvtChangedActive
- WinEvtChangedChild
- WinEvtChangedDesign
- WinEvtClicked
- WinEvtClose
- WinEvtCreated
- WinEvtCrxNavigate
- WinEvtCtxEvent
- WinEvtDbFldUpdate
- WinEvtDragInit
- WinEvtDragTerm
- WinEvtDrop
- WinEvtDropEnter
- WinEvtDropLeave
- WinEvtDropOver
- WinEvtEndSession
- WinEvtFocusCancel
- WinEvtFocusInit
- WinEvtFocusTerm
- WinEvtFsiMonitor
- WinEvtHelpTip
- WinEvtHyphenate
- WinEvtInit
- WinEvtIvlDropItem
- WinEvtJob
- WinEvtKeyItem
- WinEvtLstDataInit
- WinEvtLstEditActivate
- WinEvtLstEditCommit
- WinEvtLstEditEndGroup

Kontakt

- WinEvtLstEditEndItem
- WinEvtLstEditFinished
- WinEvtLstEditStart
- WinEvtLstEditStartGroup
- WinEvtLstEditStartItem
- WinEvtLstGroupArrange
- WinEvtLstGroupInit
- WinEvtLstRecControl
- WinEvtLstSelect
- WinEvtLstSelectRange
- WinEvtLstViewInit
- WinEvtMdiActivate
- WinEvtMenuCommand
- WinEvtMenuContext
- WinEvtMenuInitPopup
- WinEvtMenuPopup
- WinEvtMouse
- WinEvtMenuItem
- WinEvtMouseMove
- WinEvtNodeExpand
- WinEvtNodeSearch
- WinEvtNodeSelect
- WinEvtPageSelect
- WinEvtPosChanged
- WinEvtReadOnlyChanged
- WinEvtSocket
- WinEvtTapi
- WinEvtTerm
- WinEvtTimer
- WinEvtUser

In (alpha2) wird der Prozedur- und der Funktionsname übergeben. Prozedur- und Funktionsname werden dabei durch einen : voneinander getrennt.

Konnte das Ereignis nicht gesetzt werden, ist der Rückgabewert false. In diesem Fall verfügt das Objekt nicht über das angegebene Ereignis. Bei erfolgreicher Durchführung ist der Rückgabewert true.



Bei dynamisch erzeugten Objekten können die Ereignisse EvtInit und EvtTerm nicht gesetzt werden.

Beispiel:

```
tButton->WinEvtProcNameSet(_WinEvtClicked, 'Proc:EvtClicked');
```

Sonstige Befehle

Sonstige Befehle

Liste sortiert

nach

Gruppen,

Alphabetische

Liste aller

Befehle

- ColorMake
- FontMake
- PointMake
- RangeMake
- RectMake
- RtfTabMake
- WinBarcodeSaveImage
- WinBeep
- WinBoxScrollVisible
- WinColorOpacityGet
- WinColorOpacitySet
- WinFlash
- WinHalt
- WinIconPreload
- WinLayer
- WinLstEdit
- WinPicSaveImage
- WinSearch
- WinSearchClear
- WinSearchPath
- WinSearchPathGet
- WinShutdownBlock
- WinUpdate
- WinWbnExecCommand

Konstanten

- WinBeepDefault
- WinBeepError
- WinBeepInformation
- WinBeepQuestion
- WinBeepWarning
- WinDialogMaximized
- WinDialogMinimized
- WinDialogNormal
- WinDialogPrimary
- WinFlashCaption
- WinFlashDefault
- WinFlashStop
- WinFlashTray
- WinFlashUntilFore

- [WinFlashUntilStop](#)
- [WinGntRecalc](#)
- [WinGntRefresh](#)
- [WinImageFormatBmp](#)
- [WinImageFormatJpg](#)
- [WinImageFormatPng](#)
- [WinImageFormatTif](#)
- [WinLayerDarken](#)
- [WinLayerEnd](#)
- [WinLayerStart](#)
- [WinLayerUpdate](#)
- [WinLstActiveOnly](#)
- [WinLstEditClearChanged](#)
- [WinLstEditLst](#)
- [WinLstEditLstAlpha](#)
- [WinLstFromFirst](#)
- [WinLstFromLast](#)
- [WinLstFromSelected](#)
- [WinLstFromTop](#)
- [WinLstIgnoreHidden](#)
- [WinLstPosBottom](#)
- [WinLstPosMiddle](#)
- [WinLstPosSelected](#)
- [WinLstPosTop](#)
- [WinLstRecDoSelect](#)
- [WinLstRecEvtSkip](#)
- [WinLstRecFromBuffer](#)
- [WinLstRecFromRecId](#)
- [WinLstReorderColumns](#)
- [WinMsgAll](#)
- [WinMsgNoInput](#)
- [WinMsgNoKeyboardInput](#)
- [WinMsgNoMouseInput](#)
- [WinPicSignReset](#)
- [WinPicSignSaveAsExt](#)
- [WinPicSignSaveAsJpg](#)
- [WinPicSignSaveAsPng](#)
- [WinPicSignSaveAsTif](#)
- [WinPicSignSavePic](#)
- [WinPicSignSaveSign](#)
- [WinRvwEditAbortEditing](#)
- [WinRvwUpdateDoKeepSelect](#)
- [WinRvwUpdateDoSelect](#)
- [WinRvwUpdateFromFirst](#)
- [WinRvwUpdateFromLast](#)
- [WinRvwUpdateFromRecBuf](#)
- [WinRvwUpdateFromSelected](#)
- [WinRvwUpdateFromTop](#)
- [WinRvwUpdateFromTopLocation](#)
- [WinRvwUpdateOptClearCache](#)
- [WinRvwUpdateOptClearSelected](#)

Kontakt

- [WinUpdActivate](#)
- [WinUpdBuf2Obj](#)
- [WinUpdFld2Buf](#)
- [WinUpdFld2Obj](#)
- [WinUpdObj2Buf](#)
- [WinUpdObj2Fld](#)
- [WinUpdOff](#)
- [WinUpdOn](#)
- [WinUpdSort](#)
- [WinUpdState](#)
- [WinWbnCommandExecPrint](#)
- [WinWbnCommandNavBack](#)
- [WinWbnCommandNavForward](#)
- [WinWbnCommandNavHome](#)
- [WinWbnCommandNavSearch](#)
- [WinWbnCommandSelCopy](#)
- [WinWbnCommandSelCut](#)
- [WinWbnCommandSelDelete](#)
- [WinWbnCommandSelPaste](#)
- [WinWbnCommandSelSelectAll](#)
- [WinWbnCommandUpdNoCache](#)
- [WinWbnCommandUpdNormal](#)
- [WinWbnCommandZoomIn](#)
- [WinWbnCommandZoomOut](#)
- [WinWbnCommandZoomReset](#)

Kontakt

ColorMake(int1[, byte2]) : color Farbe

erzeugen int1

Farbwert

Transparenz
byte2
(optional)

Resultat color Farbe

Verwandte

Siehe Befehle, color,
ColorRgbMake()

Mit diesem Befehl kann aus einem Farbwert (int1) und einer Transparenz (byte2) eine Farbe zusammengesetzt werden. In (int1) kann ein Farbwert oder eine der _WinCol...-Konstanten angegeben werden. Der Befehl setzt die Eigenschaften der Farbe. Wird keine Transparenz angegeben, ist die Farbe deckend (Transparenz = 0).



Der Befehl ignoriert eine evtl. durch WinColorOpacitySet() gesetzte Deckkraft im Farbwert des Argumentes (int1).

Beispiele:

```
tColor # ColorMake(_WinColWhite, 128)      // Weiß mit 50 Transparenz
```

Mögliche Laufzeitfehler:

ErrValueInvalid In (int1) ist ein ungültiger Farbwert übergeben worden.

Kontakt

ColorRgbMake(byte1,  byte2, byte3) : int

Farbwert erzeugen byte1

Farbanteil rot

farbanteil

byte2

grün

Farbanteil

byte3

blau

Resultat int Farbwert

Verwandte Befehle,

Siehe

color,

ColorMake()

Mit diesem Befehl kann aus den drei Farbanteilen (rot, grün, blau) ein Farbwert erstellt werden (byte1 = rot, byte2 = grün, byte3 = blau). Der daraus resultierende Farbwert kann in eine der Farbeigenschaften geschrieben werden.

Beispiele:

```
ColorRgbMake(255, 255, 255) // weiß ColorRgbMake(255, 0, 0) // hellrot
```

Kontakt

FontMake(alpha1[, int2[, int3]]) :



font

Zusammenstellen eines font-Wertes

Name der

alpha1

Schriftart

Größe der

int2 Schriftart

(optional)

Schrift-Attribute

int3

(optional)

Resultat font Schriftart

font (Datentyp),

Siehe Font

(Eigenschaft)

Dieser Befehl erzeugt aus den Angaben eine font-Struktur. Im Parameter (alpha1) wird der Name der Schriftart angegeben. Wird eine leere Zeichenkette übergeben wird die Schriftart des Systems verwendet.

In den Parametern (int2) und (int3) können die Größe der Schriftart in 1/10 Punkten und die Schriftattribute angegeben werden. Für die Schriftattribute stehen folgende Konstanten zur Verfügung:

WinFontAttrNormal Normale Darstellung

WinFontAttrBold Fett

WinFontAttrItalic Kursiv

WinFontAttrStrikeOut Durchgestrichen

WinFontAttrUnderline Unterstrichen

Die Schrift-Attribute können miteinander kombiniert werden.

Beispiele:

```
// Arial 10ptFont # FontMake('Arial', 100, _WinFontAttrNormal); // System font 12pt bold and unde
```

Mögliche Laufzeitfehler:

In (alpha1) wurde eine Zeichenkette länger als 31 Zeichen

ErrStringOverflow übergeben.

ErrValueInvalid Im Parameter (int3) wurde ein ungültiger Wert übergeben.

Kontakt

PointMake(int1, int2) :



point

Erzeugung eines Point

int1 X-Wert

int2 Y-Wert

Resultat point Erzeugter Punkt

Verwandte Befehle,

Siehe point

Dieser Befehl erzeugt durch die Angaben eines Punktes eine point-Struktur.

Beispiel:

```
local{  tArea          : point; }  ...  // Erzeugen eines Punktes auf 100, 100  tArea = PointMake(1
```

Kontakt

RangeMake(int1, int2) :



range

Erzeugung einer Markierung

int1 Anfang

int2 Ende

Resultat range Bereich

Siehe range, Range

Dieser Befehl erzeugt aus den Angaben des Bereichsanfangs und -endes eine range-Struktur.

Der Bereichsanfang und -ende wird in Zeichen angegeben. Bei den Eingabe-Objekten kann der komplette Text mit RangeMake(0, -1) selektiert werden.

Kontakt

**RectMake(int1, int2, int3,
int4) : rect Erzeugung
eines Rects**



int1	Randes Abstand des linken
int2	Randes Abstand des oberen
int3	Randes Abstand des rechten
int4	Randes Abstand des unteren

Resultat rect Erzeugtes Rechteck

Siehe Verwandte Befehle, rect

Dieser Befehl erzeugt aus den Angaben des Anfangs- und Endpunktes eine rect-Struktur.

Beispiel:

```
local{ tArea : rect; } ... // Erzeugen eines Rechteck auf 100, 100 mit der Größe 100, 100 tAr
```

Kontakt

WinBeep([int1])  Systemklang 

wiedergeben
int1 Systemklang

(optional)
Verwandte

Siehe [SysBeep\(\)](#),

[Benutzerdefinierte Sounds \(Blog\)](#)

Mit diesem Befehl kann ein Systemklang wiedergegeben werden. Folgende Konstanten sind zulässig:

WinBeepDefault Systemklang für "Standardton Warnsignal"

WinBeepInformation Systemklang für "Sternchen"

WinBeepError Systemklang für "Kritischer Fehler"

WinBeepWarning Systemklang für "Hinweis"

WinBeepQuestion Systemklang für "Frage"

Wird kein Argument angegeben, wird WinBeepDefault verwendet.

Mögliche Laufzeitfehler:

ErrValueInvalid In (int1) wurde kein gültiger Wert angegeben.

Kontakt

obj -> WinBoxScrollVisible(handle1) :



logic

Scrollt auf ein nicht sichtbares Objekt

obj Scrollbox-Objekt

Objekt, das in den

handle1 Anzeigebereich gescrollt

werden soll

Resultat logic Scrollen notwendig

Siehe Verwandte Befehle

Dieser Befehl scrollt das Oberflächenobjekt (handle1) in den Anzeigebereich der **Scrollbox** (obj), sofern es nicht bereits im sichtbaren Bereich ist.

Der Rückgabewert gibt Aufschluss darüber, ob ein Scrolling notwendig war (**true**) oder nicht (**false**).

Mögliche Laufzeitfehler:

ErrHdIInvalid In (obj) wurde kein Deskriptor eines **Scrollbox**-Objektes angegeben oder (handle1) ist kein untergeordnetes Objekt der **Scrollbox**.

Kontakt

WinColorOpacityGet(int1) : int 

Deckkraft einer Farbe ermitteln

int1 Farbwert

Resultat int Deckkraft in Prozent

Verwandte Befehle,

Siehe

WinColorOpacitySet()

Dieser Befehl ermittelt die Deckkraft eines Farbwertes (int1). Die Deckkraft (int2) wird mit Werten zwischen 0 (transparent) und 100 (deckend) angegeben.

Als Farbwert (int1) kann ein beliebiger RGB-Wert (Siehe ColorRgbMake()) oder eine der _WinCol-Farbkonstanten (z. B. WinColLightRed) angegeben werden. Bei nicht erlaubten Farbwerten (WinColUndefined, WinColParent, WinColTransparent) und Systemfarben (z. B. WinColScrollBar) wird der Laufzeitfehler ErrValueInvalid ausgelöst.

Beispiel:

```
// Deckkraft von rot ermittelntOpacity # WinColorOpacityGet(_WinColLightRed); // Deckkraft einer R
```

Mögliche Laufzeitfehler:

ErrValueInvalid Ungültiger Farbwert (z. B. WinColUndefined, WinColParent, WinColTransparent, WinColScrollBar) angegeben.

Kontakt

**WinColorOpacitySet(int1,
int2) : int**



Deckkraft einer Farbe setzen

int1 Farbwert

int2 Deckkraft in Prozent

Resultat int Um Deckkraft erweiterter

Farbwert

Verwandte Befehle,

Siehe

WinColorOpacityGet()

Dieser Befehl setzt die Deckkraft (int2) eines Farbwertes (int1) und gibt den neuen Farbwert zurück.
Die Deckkraft (int2) kann mit Werten zwischen 0 (transparent) und 100 (deckend) angegeben
werden.

Als Farbwert (int1) kann ein beliebiger RGB-Wert (Siehe [ColorRgbMake\(\)](#)) oder eine der [_WinCol-](#)
Farbkonstanten (z. B. [_WinColLightRed](#)) angegeben werden. Handelt es sich bei der Farbkonstante
um eine Systemfarbe ([_WinColScrollBar](#) etc.), dann wird die Systemfarbe durch den entsprechenden
RGB-Wert ersetzt. Bei nicht erlaubten Farbwerten ([_WinColUndefined](#), [_WinColParent](#),
[_WinColTransparent](#)) wird der Laufzeitfehler [_ErrValueInvalid](#) ausgelöst.

Ist das Argument Opacity < 0 oder > 100, wird es entsprechend in den Bereich 0 ...
100 verschoben.

Ist in einem Farbwert bereits ein Deckkraft gesetzt, wird diese durch den neuen Wert überschrieben.

Beispiel:

```
// Rote Farbe mit 50 % Deckkraft setztentColor # WinColorOpacitySet(_WinColLightRed, 50); // RGB-Fa
```

Mögliche Laufzeitfehler:

ErrValueInvalid Ungültiger Farbwert (z. B. [_WinColUndefined](#), [_WinColParent](#),
[_WinColTransparent](#)) angegeben.

Kontakt

obj ->



WinCroNavigate(int1)

Navigiert zu einer URL
Objekt

obj

(Chromium-Objekt)

int1 Art der Navigation

Siehe Verwandte Befehle

Navigiert zu einer URL, deren Inhalt im Verlauf der Navigation bereits angezeigt wurde, falls eine solche Navigation stattgefunden hat.

Im Parameter (int1) können folgende Optionen übergeben werden:

- _WinCroNavBack (1)

Navigiert zur vorhergehenden Seite.

- _WinCroNavForward (2)

Navigiert zur nächsten Seite.

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

ErrValueInvalid In Argument (int1) wurde keiner der gültigen Werte für die Navigation angegeben.

Kontakt

obj ->



WinCroPrint()

Inhalt drucken

 Objekt

obj

 ([Chromium-Objekt](#))

Siehe [Verwandte Befehle](#)

Der Befehl ruft einen Dialog zum Drucken des Inhaltes der aktuell angezeigten Seite aus.

Mögliche Laufzeitfehler

ErrHdIInvalid Bei (obj) handelt es sich nicht um ein [Chromium-Objekt](#).

Kontakt

obj -> WinCroReload(int1)



Lädt den Inhalt der aktuellen URL erneut

 Objekt

 obj

(Chromium-Objekt)

 int1 Optionen

Siehe Verwandte Befehle

Der Befehl lädt den Inhalt der aktuell angezeigten Seite erneut. Über Optionen kann gesteuert werden, ob die Seite neu angefordert oder aus dem Cache geladen werden soll.

Im Parameter (int1) können folgende Optionen übergeben werden:

- _WinCroReloadIgnoreCache (1)

 Zwischengespeicherte Inhalte werden erneut angefordert.

- _WinCroReloadNormal (0)

 Zwischengespeicherte Inhalte werden nicht erneut angefordert (default).

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

ErrValueInvalid In Argument (int1) wurde keine der gültigen Optionen angegeben.

Kontakt

obj -> WinCroSelection(int1) Aktionen für

den ausgewählten Inhalt

Objekt

(Chromium-Objekt)

int1 Aktion

Siehe Verwandte Befehle

Der Befehl führt Aktionen auf den aktuell angezeigten Inhalt aus, sofern möglich.

Im Parameter (int1) können folgende Optionen übergeben werden:

- WinCrosSelCopy (1)

Kopiert den ausgewählten Inhalten in die Zwischenablage.

- WinCrosSelCut (2)

Schneidet den ausgewählten Inhalt in die Zwischenablage aus.

- WinCrcSelDelete (4)

Löscht den ausgewählten Inhalt.

- **WinCroSelPaste (3)**

Ersetzt den ausgewählten Inhalt durch den Inhalt in der Zwischenablage.

- **WinCroSelSelectAll (5)** Wählt den

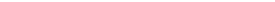
gesamten Inhalt aus.

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein Chromium-Objekt.

ErrValueInvalid In Argument (int1) wurde keine der gültigen Optionen angegeben.

Kontakt

obj -> WinFlash([int1])  Anwendung in der

Taskleiste blinken lassen

obj Deskriptor des Fensters

Optionen (optional)	Lässt die
<u>WinFlashDefault</u>	Titelzeile und das Icon in der Taskleiste blinken.

[WinFlashTray](#) Lässt das Icon
in der Taskleiste

_WinFlashCaption blinken.
Lässt die
Titelzeile der
Anwendung
blinken.

int1 WinFlashUntilStop **Fenster**
blinken lassen,
bis
WinFlashStop
aufgerufen
oder das
Fenster
beendet wird.

WinFlashUntilFore Fenster blinken lassen, bis es in den Vordergrund geholt wird.

[WinFlashStop](#) **Blinken des Fensters stoppen.**

Siehe Verwandte Befehle

Mit diesem Befehl kann der Anwender darauf hingewiesen werden, dass die Anwendung seine Aufmerksamkeit erfordert. Im Regelfall wird der Befehl verwendet, wenn die Anwendung nicht im Vordergrund oder minimiert ist.

Der Deskriptor des Fensters, dass die Aufmerksamkeit des Anwenders erfordert, wird in (obj) angegeben. Folgende Konstanten können in (int1) angegeben werden:

WinFlashDefault Lässt die Titelzeile und das Icon in der Taskleiste blinken.

WinFlashTray Lässt das Icon in der Taskleiste blinken

WinFlashCaption Lässt das Icon in der Taskleiste blinken.

WinFlashCaption Lässt die Titelleiste der Anwendung blinken.
WinFlashUntilStop Fenster blinken lassen, bis **WinFlashStop** aufgerufen oder das Fenster beendet wird

WinFlashUntilForeground Fenster blinken lassen, bis es in den Vordergrund geholt wird.

WinFlashStop Blinken des Fensters stoppen.

Kontakt

Die Optionen WinFlashDefault, WinFlashTray und WinFlashCaption können mit einer der Optionen WinFlashUntilStop bzw. WinFlashUntilFore kombiniert werden.

Ist weder die Option WinFlashUntilStop noch WinFlashUntilFore angegeben, bewirkt dies ein dreimaliges Aufblitzen.

Wird (int1) nicht angegeben, wird WinFlashDefault verwendet.



Ist (obj) der Deskriptor eines MdiFrame, wirkt sich die Option WinFlashTray nicht aus.

Mögliche Laufzeitfehler:

ErrHdlInvalid Im Argument (obj) wurde kein gültiger Deskriptor eines Fenster-Objektes

ErrValueInvalid In (int1) wurde kein gültiger Wert angegeben.

Kontakt

WinHalt() CONZEPT 16-Client beenden / Prozedur
beenden

Siehe [Alle Befehle](#)

Der Befehl WinHalt() beendet die aktuelle Prozedur und den gesamten CONZEPT 16-Client.
Geladene Dialoge müssen zuvor über [WinClose\(\)](#) geschlossen werden.

Sofern die Prozedur im Designer gestartet worden ist, wird nur die Prozedur abgebrochen
und der Designer wieder aktiviert.



Der Befehl hat keine Auswirkung auf den Hauptprozess, wenn er in einem, per [JobStart\(\)](#)
gestarteten, Task bzw. Prozess ausgeführt wird.

Kontakt

WinIconPreload(alpha1) :



int

Icons in den Cache laden

alpha1 Zu ladende Icons

Resultat int Anzahl geladener Icons

Siehe Icon

Normalerweise werden anzuzeigende Icons erst direkt vor der Anzeige im Dialog vom Datenbankserver geladen. Werden viele Icons auf einmal geladen, kann dies zu einem sichtbaren Aufbau der Anzeige führen. Mit diesem Befehl können darzustellende Icons direkt in den Icon-Cache geladen werden. Die Icons müssen dann später bei der Anzeige nicht mehr vom Datenbank-Server abgeholt werden, da diese dann bereits im Cache vorliegen.



Der Cache speichert maximal 2000 Icons. Ist der Cache vollständig gefüllt kehren weitere Aufrufe mit dem Resultat 0 zurück.

Im Argument (alpha1) wird dem Befehl eine kommasseparierte Liste der zu ladenden Icon-Namen übergeben. Diese Liste kann auch die Wildcard-Zeichen '*' und '?' enthalten.

Der Rückgabewert gibt die Anzahl der geladenen Icons an.

Beispiele:

```
// Icons mit den Namen Icon1, Icon2 und Icon3 in den Icon-Cache laden  
# WinIconPreload('Icon1,Icon2,Icon3')
```

Wird der Befehl mit einer leeren Zeichenkette aufgerufen, wird der Icon-Cache geleert. Dies kann sinnvoll sein, da die im Cache vorliegenden Icons bis zur Beendigung des Clients im Cache verbleiben.

Kontakt

WinLayer(int1[, handle2[, int3[,
alpha4[, int5]]]]) : int Layer-Modus
für Frame-Objekte int1 Layer-Modus



handle2 Frame-Deskriptor

int3 Anzeigedauer

alpha4 Hinweistext

int5 Layer-Optionen

Resultat der Durchführung

ErrOK Der Befehl

wurde

erfolgreich

durchgeführt

ErrInUse Der Befehl

wurde

wiederholt mit



WinLayerStart

aufgerufen

ErrUnavailable Der Befehl kann

nicht

durchgeführt

werden (siehe

Text)

Siehe Verwandte Befehle

Der Befehl "friert" die Ausgabe eines vorgegebenen Frame-Objektes für eine bestimmte Zeitspanne ein. Änderungen an untergeordneten graphischen Objekten, während dieser Zeitspanne werden zwar durchgeführt, jedoch erst sichtbar nachdem die Zeitspanne abgelaufen ist, bzw. Layer-Modus abgeschaltet wird.

Für den Layer-Modus (int1) kann eine der folgenden Konstanten angegeben werden:

- WinLayerStart

Diese Option startet den Layer-Modus für das Frame-Objekt (handle2). Es kann sich zu einem bestimmten Zeitpunkt jeweils nur ein Frame im Layer-Modus befinden. Das Argument (int3) definiert die Zeitspanne in Millisekunden, bis zur Beendigung des Layer-Modus. Wird das Argument nicht übergeben oder ist es 0, wird eine Zeitspanne von einer Sekunde angenommen. Im Argument (alpha4) kann eine Meldung übergeben werden, die in der Mitte des Anwendungsfensters ausgegeben wird. So kann der Benutzer über den Vorgang informiert werden.

- WinLayerEnd

Diese Option beendet den Layer-Modus vor Ablauf der Zeitspanne. Es darf kein Frame-Deskriptor (handle2) angegeben werden. Alle anderen Argumente werden ignoriert.

- WinLayerUpdate

Wird das Frame-Objekt während des Layer-Modus in Position oder Größe

Kontakt

verändert, kann der Layer durch Aufruf mit dieser Option an das neue Fensterrechteck angepasst werden. Es darf kein Frame-Deskriptor (handle2) angegeben werden. Die übrigen Argumente können angegeben werden, um eine Änderung der Zeitspanne (int3) oder der Meldung (alpha4) durchzuführen. Das Argument (int5) wird ignoriert.

Im Argument (int5) können beim Modus (int1) WinLayerStart folgende Optionen angegeben werden:

WinLayerDarken Fenster abdunkeln

WinLayerDCross Fenster mit Kreuz-Schraffur überdecken Diese Optionen können kombiniert werden.

War die Durchführung erfolgreich, liefert der Befehl ErrOk. Im Fehlerfall ist das Resultat ein negativer Wert. Wird der Fehlerwert zurückgegeben, liegt einer der folgenden Fälle vor:

- Es gibt keine Fensterrepräsentation des Frame auf dem Bildschirm. Dies ist z. B. im Ereignis EvtInit der Fall. Die Fensterrepräsentation wird durch WinCreate() bzw. WinDialogRun() erzeugt.
- Der Befehl wurde mit WinLayerStart / WinLayerEnd aufgerufen, es existiert jedoch kein Layer, entweder weil zuvor nicht WinLayerStart durchgeführt wurde oder weil der Layer bereits beendet wurde.

Beispiele:

```
// Während des Theme-Wechsels, Hinweis für 1 Sekunde aktivierenWinLayer(_WinLayerStart, $Frame, 1)
```

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der Deskriptor (handle2) bezeichnet kein Frame-Objekt. Die angegebene Zeitspanne (int3) ist negativ oder größer als
<u>ErrRange</u>	<u>MaxInt</u> Millisekunden.
<u>ErrValueInvalid</u>	Der Wert für (int5) ist ungültig.

Kontakt

obj ->

WinPicSaveImage(alpha1      handle2[, int3[, int4[, int5]]]) : int

Bildinhalt speichern
Deskriptor des

obj

Picture-Objektes
Dateiname der

externen Datei
Deskriptor eines

handle2 Memory-Objektes

(optional)
Optionen

int3

(optional)
Hintergrundfarbe

int4 der Zeichnung

(optional)
Qualitätsstufe

int5

(optional)

Resultat int Fehlercode 

Verwandte

Siehe Befehle,

SignMode

Dieser Befehl speichert die Zeichnung aus dem Picture-Objekt (handle) in der Datei (alpha1) oder einem Memory-Objekt (handle2). Wird im Argument (handle2) ein Wert ungleich 0 angegeben, wird der Dateiname (alpha1) ignoriert.

Das Argument (int3) definiert das Dateiformat und die zu speichernden Inhalte.

Folgende Konstanten können angegeben werden:

- WinPicSignSaveAsExt - Format wird anhand des Dateinamens (alpha1) bestimmt. Dies ist nur möglich, wenn kein Memory-Objekt (handle2) angegeben wurde.
- WinPicSignSaveAsJpg - Zeichnung im JPEG-Format speichern.
- WinPicSignSaveAsPng - Zeichnung im PNG-Format speichern.
- WinPicSignSaveAsTif - Zeichnung im TIFF-Format speichern.
- WinPicSignSavePic - Inhalt des Picture-Objektes wird gespeichert.
- WinPicSignSaveSign - Zeichnung wird gespeichert.
- WinPicSignReset - Zeichnung wird gelöscht und nur das originale Bild im Picture-Objekt angezeigt.

Die Optionen WinPicSignSavePic, WinPicSignSaveSign und WinPicSignReset können miteinander und je einer der WinPicSignSaveAs...-Optionen kombiniert werden.

Wird das Bild extern als Datei gespeichert und die Option WinPicSignSaveAsExt ist nicht angegeben, wird die Dateiendung des ausgewählten Formats dem Dateinamen hinzugefügt.

Kontakt

Ist keine der Optionen WinPicSignSavePic und WinPicSignSaveSign angegeben, wird auch kein Bild erzeugt. Wird hingegen das Argument (int3) nicht angegeben oder 0 übergeben, werden die Optionen WinPicSignSaveAsJpg, WinPicSignSavePic und WinPicSignSaveSign kombiniert.

Im Argument (int4) kann ein Farbwert (siehe _WinCol... übergeben werden, mit dem der Hintergrund der Zeichnung gefüllt wird, wenn die Option WinPicSignSavePic nicht angegeben ist. Wird WinColUndefined angegeben, oder das Argument weggelassen, wird die Hintergrundfarbe vom Picture-Objekt verwendet. Für PNG- und TIFF-Dateien kann auch WinColTransparent angegeben werden.

Das Argument (int5) definiert die Qualität des Bildes, wenn die Option

WinPicSignSaveAsJpg oder die Option WinPicSignSaveAsExt in Kombination mit dem Dateityp '.jpg' angegeben ist. Es können Werte zwischen 1 und 100 verwendet werden. Standardmäßig wird eine Qualitätsstufe von 90 Prozent angewendet.

Resultat

Die Funktion gibt ErrOk zurück, wenn die Zeichnung erfolgreich gespeichert wurde.

Zusätzlich können folgende Fehlercodes zurückgegeben werden:

ErrGeneric

Allgemeiner Fehler aufgetreten.

ErrFsiOpenFailed

Externe Datei kann nicht geöffnet werden.

ErrFsiWriteFault

Externe Datei kann nicht geschrieben werden.

ErrUnavailable

Der Zeichnungsmodus ist nicht aktiv, die Dateiendung ist unbekannt oder es ist keine _WinPicSignSaveAs...-Konstante angegeben.

ErrMemExhausted Speicher nicht ausreichend.

Beispiel:

```
// Zeichnung inklusive Hintergrundbild als Datei Sign.jpg im temporären Verzeichnis speichern $p
```

Mögliche Laufzeitfehler:

ErrHdIInvalid Deskriptor des Picture-Objektes (obj) oder des Memory-Objektes (handle2) ist ungültig.

Kontakt

obj -> WinSearch(alpha1) : handle  Suchen

eines Objekts über den Namen

obj Startobjekt der Suche
Name des zu suchenden

alpha1 Objekts

Resultat handle Deskriptor des Objekts

Siehe Verwandte Befehle, Blog

Liefert den Deskriptor des gesuchten Objekts. Die Wildcard-Operatoren '*' und '?' können im Objektnamen angegeben werden und werden bei der Suche entsprechend berücksichtigt. Als Startobjekt wird das Eltern-Objekt (Frame, GroupBox ...) des zu suchenden Objekts (alpha1) angegeben.

Als Resultat wird der Deskriptor des gefundenen Objekts zurückgegeben. Wurde kein Objekt gefunden ist das Resultat 0.



Unterobjekte von ToolbarMenu-Objekten können aus Kompatibilitätsgründen nicht gefunden werden. Siehe Kompatibilitätshinweise bei ToolbarMenu.

Beispiel:

```
$Form->WinSearch('rlsAdress'); $Form->WinSearch('rls*')
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

WinSearchClear()  **Löschen des Suchpfades für Objekte**

Verwandte

Siehe [Befehle](#),

[WinSearchPath\(\)](#)

Mit diesem Befehl wird ein mit [WinSearchPath\(\)](#) gesetzter Suchpfad gelöscht.

Kontakt

obj -> WinSearchPath() Setzen des
Suchpfades für Objekte
Startobjekt des
obj

Suchpfades

Verwandte

Siehe Befehle,

WinSearchPathGet(),

WinSearchClear()



Werden innerhalb von Prozeduren Referenzen auf Objekte mit dem Objektnamen (\$Name) verwendet, wird der Suchpfad benutzt, um das Objekt mit dem Namen im Objektbaum zu finden und den entsprechenden Deskriptor zu ermitteln. Die Suche wird dabei von einem bestimmten Objekt innerhalb des Suchbaums gestartet und durchsucht alle untergeordneten Objekte. Wird das Objekt nicht gefunden, werden auch die anderen Bäume durchsucht. Die Suche wird beendet, sobald das erste Objekt mit dem entsprechenden Namen gefunden wird.

Dieses Startobjekt bestimmt wie lange die Suche dauert, wenn das Objekt gefunden werden kann. Der Suchpfad wird automatisch gesetzt, wenn ein Ereignis ausgelöst oder ein Frame-Objekt geladen wird. Alle Namensreferenzen werden dann innerhalb des Fenster-Objekts aufgelöst.

In einer Applikation mit MDI-Fenstern wird nicht nur das aktive Fenster durchsucht, sondern auch alle anderen MDI-Fenster, wenn das angegebene Objekt nicht im aktiven MDI-Fenster gefunden wurde. Dies verlängert die Suche. Die Suche kann über das Flags WinAppSearchMdiFrame des Applikation-Objekts auf das aktuelle MDI-Fenster beschränkt werden. Der Suchpfad muss dann nicht gesetzt werden.

Mit dem Befehl wird das Startobjekt des Suchpfades definiert. Wurde innerhalb eines Ereignisses ein Objekt mit dem Befehl WinOpen() mehrfach geladen, ist der Suchpfad zunächst auf das zuletzt geladene Fenster gesetzt. Da die Namen der Objekte nicht mehr eindeutig sind, muss, um ein Objekt aus dem zuerst geladenen Fenster anzusprechen, der Suchpfad auf dieses Fenster gesetzt werden.

Durch die Angabe eines Suchpfades wird das Startobjekt (obj) für die Suche festgelegt. Bei der Suche werden ebenfalls die Unterobjekte des Startobjektes durchlaufen. Im Beispiel werden zwei Dialoge geladen, die beide ein Objekt mit dem Namen "Text" besitzen. Die Identifizierung des Objektes mit \$Text ist nicht eindeutig. Durch die Angabe eines Suchpfades mit dem entsprechenden Dialog wird das Objekt eindeutig gefunden.

Nachdem die Objekte gefunden wurden, muss der Suchpfad mit der Anweisung WinSearchClear() wieder zurückgesetzt werden.

Beispiel:

```
WinOpen('Meldung1', _WinOpenDialog);WinOpen('Meldung2', _WinOpenDialog);$Meldung1->WinSearchPath(
```

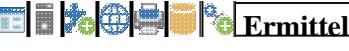


Namensreferenzen innerhalb von Prozeduren werden für jeden Aufruf nur einmal aufgelöst. D.h. wurde in dem obigen Beispiel der Deskriptor des Objektes Text bereits zuvor ermittelt, wird unabhängig vom gesetzten Suchpfad, in der Folge der Prozedur immer das gleiche Objekt angesprochen. Soll ein anderes Objekt

Kontakt

mit gleichen Namen angesprochen werden, muss der Deskriptor des Objektes mit dem Befehl WinSearch() ermittelt werden.

Kontakt

WinSearchPathGet() : handle |  Ermitteln des Suchpfades für Objekte

Resultat handle Deskriptor des Objektes

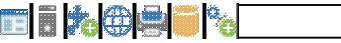
Siehe Verwandte Befehle,
WinSearchPath(),
WinSearchClear()

Mit diesem Befehl kann ein mit WinSearchPath() gesetzter Suchpfad abgefragt werden. Es wird der Deskriptor des Objektes zurückgegeben.

Beispiel:

```
WinOpen('Message1', _WinOpenDialog);WinOpen('Message2', _WinOpenDialog);tHdlSearchPath # WinSearc
```

Kontakt

obj -> WinShutdownBlock([alpha1]) : int; 

Beenden der Windows-Sitzung verhindern

obj Deskriptor des Fenster-Objekts

alpha1 Blockierungsgrund (optional)

Fehlerwert

ErrOk

Kein Fehler aufgetreten

Resultat int ErrUnavailable Die Funktion steht nicht



zur Verfügung.

ErrGeneric

Unerwarteter Fehler.

Siehe Verwandte Befehle, EvtEndSession

Mit dem neuen Befehl WinShutdownBlock kann verhindert werden, dass Windows die Sitzung beendet. Er sollte nur dann verwendet werden, wenn eine Operation in Verarbeitung ist, die abgeschlossen werden muss.



Dieser Befehl verhindert nicht das Beenden des Clients mittels Taskmanager.

Als (obj) muss der Deskriptor auf ein Fenster-Objekt (Frame, AppFrame, MdiFrame oder TrayFrame) übergeben werden. Die Fenster-Erstellung muss bereits abgeschlossen sein. Dies ist der Fall, wenn das Ereignis EvtCreated durchlaufen wurde.

Das optional Argument (alpha1) enthält die Nachricht, die im Abmeldebildschirm von Windows bei der blockierenden Anwendung angezeigt wird.



Die Nachricht sollte möglichst kurz und prägnant sein.

Fehlt das Argument (alpha1) oder ist es leer, wird die Blockierung für das Fenster (obj) wieder aufgehoben.

Der Rückgabewert der Anweisung kann mit folgenden Konstanten verglichen werden:

ErrOk Der Befehl wurde erfolgreich durchgeführt.

ErrUnavailable Die Funktion steht nicht zur Verfügung. Für die Ausführung wird Windows Vista vorausgesetzt.

ErrGeneric Es ist ein unerwarteter Fehler aufgetreten.

Beispiele:

```
// Beenden blockieren$Frame->WinShutdownBlock('Datenspeicherung abschließen');// Operation durchf
```

Mögliche Laufzeitfehler:

Der in (obj) übergebene Deskriptor ist ungültig, ist kein

ErrHdlInvalid Fenster-Objekt oder die Blockierung für das Fenster-Objekt wurde bereits aufgehoben.

Kontakt

WinSleep(int1[, int2])

Prozedurverarbeitung anhalten

int1 Anhaltedauer

int2

Nachrichtenmod

Verwandte Befehle,

Siehe WinHalt(),

SysSleep()

Diese Funktion hält die Prozedurausführung für die in (int1) angegebene Zeitspanne in Millisekunden an.

Im Gegensatz zu dem Befehl SysSleep() wird die Nachrichtenschleife von Windows abgefragt und neu eingetroffene Ereignisse bearbeitet. Ist die Zeitspanne abgelaufen oder keine Windows-Nachricht vorhanden, kehrt der Befehl zurück und setzt die Prozedurausführung fort.

Über das optionale Argument (int2) kann definiert werden, welche Nachrichten verarbeitet werden. Nachrichten, die nicht verarbeitet werden, verbleiben in der Nachrichtenschleife, um sie zu einem späteren Zeitpunkt verarbeiten zu können. Folgende Konstanten können angegeben werden:

WinMsgAll

Es werden alle von Windows gesendeten Nachrichten verarbeitet (default).

_WinMsgNoKeyboardInput Es werden keine Tastatureingabe-Nachrichten verarbeitet (default).

verarbeitet.

_WinMsgNoMouseInput Es werden keine Mauseingabe-Nachrichten verarbeitet.

WinMsgNoInput

verarbeitet.

Beispiel:

Im Ereignis EvtChanged eines Eingabe-Objektes wird während einer längeren Verarbeitung WinSleep aufgerufen, damit eingehende Windows-Nachrichten verarbeitet werden können. Hierzu zählen auch Tastatureingabe-Nachrichten. Macht der Anwender während der Durchführung des Ereignisses EvtChanged Tastatureingaben, würde dies zur Ausführung des Ereignisses EvtChanged führen. Da dieses jedoch bereits ausgeführt wird, kann die Eingabe nicht verarbeitet werden. Die Eingabe wird ignoriert und nicht an das Eingabeobjekt weitergeleitet. Durch den Aufruf von WinSleep(1, _WinMsgNoKeyboardInput) im Ereignis EvtChanged werden Tasturnachrichten ignoriert und verbleiben in der Warteschlange. Dadurch gehen keine Tastatureingaben verloren.

Kontakt

obj ->
WinUpdate([int1,
int2]) : int
Objekt aktualisieren
obj **Objekt**
 Update-Modus
int1 (siehe Text)
 Optionen für
 RecList- und
int2 **Frame-Objekte**
 (siehe Text)
Resultat int Fehlerwert
 Verwandte
Siehe Befehle,
 WinRvwUpdate()



Mit diesem Befehl wird ein Objekt aktualisiert. Sind dem übergebenen Objekt weitere Objekte untergeordnet, werden diese ebenfalls aktualisiert. Sind in einem Fenster-Objekt RecList- oder DataList-Objekte enthalten, werden diese aus Performancegründen nicht aktualisiert. Diese Objekte müssen gesondert aktualisiert werden.

Wird ein RecList-Objekt übergeben, werden alle in dem Objekt angezeigten Datensätze neu gelesen. Bei allen anderen Objekten bewirkt WinUpdate() die Anzeige der mit dem Objekt (oder dessen Unterobjekte) verknüpften Datenbankfelder. Die gleiche Wirkung wird mit WinUpdate(WinUpdFld2Obj) erzielt.

Folgende Parameter können als Update-Modus (int1) übergeben werden:

<u>WinUpdOn</u>	Setzt die <u>AutoUpdate</u> -Eigenschaft von (obj) auf true	__
<u>WinUpdOff</u>	Setzt die <u>AutoUpdate</u> -Eigenschaft von (obj) auf false	__
<u>WinUpdFld2Obj</u>	Überträgt den Inhalt der Feldpuffer in das Objekt	
<u>WinUpdObj2Fld</u>	Überträgt den Inhalt des Objektes in den Feldpuffer	
<u>WinUpdFld2Buf</u>	Überträgt den Inhalt der Feldpuffer in den Datensatzpuffer des Objekts	
<u>WinUpdBuf2Obj</u>	Überträgt den Inhalt des Textpuffers in das Objekt <u>TextEdit</u>	
<u>WinUpdObj2Buf</u>	Überträgt den Inhalt des Objektes <u>TextEdit</u> in den Textpuffer	
<u>WinUpdActivate</u>	Holt das <u>Frame</u> -Objekt in den Vordergrund	
<u>WinUpdState</u>	Status setzen	
<u>WinUpdSort</u>	<u>DataList</u> -Objekt neu sortieren	

Die Konstanten können nicht miteinander kombiniert werden. Die Konstanten

WinUpdOff und WinUpdOn können mit WinUpdScrollPos kombiniert werden, um die Scrollposition zu speichern und wieder herzustellen.

 Wird als Update-Modus (int1) WinUpdOn mit der Konstanten WinUpdScrollPos kombiniert, wird nur die Scrollposition wiederhergestellt. Zur Aktualisierung des Objektes muss WinUpdate() erneut mit WinUpdOn aufgerufen werden.

Für die Objekte RecList, DataList, Frame und GanttGraph können weitere Optionen in (int2) angegeben werden.

Kontakt

Optionen für RecList- und DataList-Objekte (in (int1) muss einer der obigen Parameter angegeben werden):

<u>WinLstFromFirst</u>	Neuaufbau der Liste ab der ersten Zeile
<u>WinLstFromLast</u>	Neuaufbau der Liste ab der letzten Zeile
<u>WinLstRecFromBuffer</u>	Neuaufbau der Liste ausgehend von den Werten des Feldpuffers
<u>WinLstRecFromRecId</u>	Neuaufbau der Liste ausgehend von der Datensatz-ID
<u>WinLstFromSelected</u>	Neuaufbau der Liste ausgehend von der selektierten Zeile
<u>WinLstFromTop</u>	Neuaufbau der Liste ab der ersten sichtbaren Zeile
<u>WinLstPosTop</u>	Selektierte Zeile wird oben in der Liste angezeigt
<u>WinLstPosMiddle</u>	Selektierte Zeile wird in der Mitte der Liste angezeigt
<u>WinLstPosBottom</u>	Selektierte Zeile wird unten in der Liste angezeigt
<u>WinLstPosSelected</u>	Selektierte Zeile behält die Position innerhalb der Liste
<u>WinLstRecDoSelect</u>	In der Liste wird der aktuelle Datensatz selektiert
<u>WinLstReorderColumns</u>	Nummeriert alle sichtbaren Spalten neu (siehe <u>ClmOrder</u>)
<u>WinLstRecEvtSkip</u>	Unterbinden des Ereignisses EvtLstSelect
<u>WinLstIgnoreHidden</u>	Die Liste wird auch aktualisiert, wenn sie unsichtbar (Visible = false) ist.
<u>WinLstActiveOnly</u>	Nur das aktive View der RecList oder DataList aktualisieren. Konstanten aus den Bereichen <u>_WinLst*From*</u> , <u>_WinLstPos*</u> , <u>WinLstReorderColumns</u> und <u>_WinLstRec*</u> können miteinander kombiniert werden. Konstanten aus gleichen Bereichen können nicht kombiniert werden.

Optionen für Frame-Objekte (in (int1) muss WinUpdState angegeben werden):

<u>WinDialogMaximized</u>	Das Fenster wird maximiert dargestellt.
<u>WinDialogMinimized</u>	Das Fenster wird minimiert dargestellt.
<u>WinDialogNormal</u>	Das Fenster wird weder maximiert noch minimiert dargestellt.
<u>WinDialogPrimary</u>	Befindet sich das Fenster außerhalb des darstellbaren Bereichs, wird es in der Bildschirmmitte des primären Bildschirms angezeigt.

Die Option WinDialogPrimary kann mit den anderen Konstanten kombiniert werden.

Optionen für GanttGraph-Objekte:

<u>WinGntRefresh</u>	Gantt-Graphen neu zeichnen
<u>WinGntRecalc</u>	Gantt-Graphen neu berechnen
Änderungen in Eingabeobjekten werden in den angegebenen Feldpuffer (Eigenschaft <u>DbFieldName</u>) übertragen, wenn der Fokus in ein anderes Objekt gesetzt wird. Diese Übertragung kann ebenfalls prozedural mit dem Parameter <u>WinUpdObj2Fld</u> durchgeführt werden.	

Die Optionen _WinLstFrom... beeinflussen nur die Anzeige der Datensätze. Der selektierte Datensatz bleibt unverändert. Die Selektion kann mit der Option WinLstRecDoSelect neu gesetzt werden. Ein Update mit dieser Option löst dann auch

Kontakt

das Ereignis EvtLstSelect der betreffenden RecList aus.

Die Position innerhalb der dargestellten Datensätze wird durch die Option _WinLstPos... bestimmt. Diese Option lässt sich mit den anderen Optionen kombinieren. Sollte eine Darstellung nicht möglich sein, weil zum Beispiel der letzte Datensatz in der Mitte dargestellt werden soll, erfolgt die Darstellung als vollständig gefüllte Liste.

Soll ein Fenster maximiert oder minimiert werden ist in (int1) die Konstante WinUpdState anzugeben.

Als Rückgabewerte wird der Fehlerwert ErrHdlInvalid zurückgegeben, wenn als (obj) kein gültiger Deskriptor angegeben wurde, sonst ErrOk.

Beispiele:

```
sub EvtLstSelect( aEvt    : event;      // Ereignis   aID      : bigint;     // Record-ID des Datensatze
```

Kontakt

obj -> WinUserEvent(word1[, int2[, alpha3]])



Benutzerdefiniertes Ereignis auslösen

obj Fenster-Objekt

word1 Ereignis-ID

 Argument des

int2 Benutzers

 (optional)

 Argument des

alpha3 Benutzers

 (optional)

Verwandte

Siehe [Befehle](#),

[EvtUser](#)

Mit dieser Anweisung wird ein Ereignis in die Ereigniswarteschlange des Betriebssystems eingefügt. Das Ereignis wird zu einem späteren Zeitpunkt aufgerufen. Zu welchem Zeitpunkt das Ereignis ausgelöst wird, kann nicht genau bestimmt werden, es wird aber nach dem Ereignis, in dem die Anweisung durchgeführt wurde, ausgelöst.

Als Objekt (obj) muss ein Fenster-Objekt übergeben werden, bei dem das Ereignis [EvtUser](#) angegeben ist. Als (int1) wird eine Benutzerdefinierte Ereignis-ID übergeben. Die ID muss im Bereich 1 - 65535 liegen und kann vom Programmierer frei gewählt werden. Die ID wird dem Ereignis übergeben und kann zur Unterscheidung mehrerer Ereignisse verwendet werden.

Die Parameter (int2) und (alpha3) können vom Benutzer frei gewählt werden. Der Inhalt wird an das Ereignis weitergereicht.

Beispiel:

```
define{ _EventUser : 10000}...tHdlFrame->WinUserEvent(_EventUser, 42, 'Answer to the ultimate qu
```

Kontakt

obj -> WinWbnExecCommand(int1)



Kommando an ein WebNavigator-Objektes senden

 Objekt

 obj (WebNavigator-Objekt)

 int1 Kommando

Siehe Verwandte Befehle

Mit diesem Befehl wird das Kommando (int1) an ein WebNavigator-Objekt gesendet.

Der Deskriptor des WebNavigator-Objektes wird in (obj) übergeben.

Im Parameter (int1) können folgende Optionen übergeben werden:

- _WinWbnCommandExecPrint (41)

 Zeigt den Druckerauswahldialog an. Bei Klick auf "OK" wird die Seite gedruckt.

- _WinWbnCommandNavHome (5)

 Navigiert zur Startseite, die in den Internet-Optionen eingestellt ist.

- _WinWbnCommandNavBack (6)

 Navigiert zur vorherigen Seite.

- _WinWbnCommandNavForward (7)

 Navigiert zur nächsten Seite.

- _WinWbnCommandNavSearch (8)

 Navigiert zur Suchseite, die im Internet-Explorer eingestellt ist.

- _WinWbnCommandSelCopy (1)

 Kopiert den aktuell selektierten HTML-Inhalt bzw. den Inhalt des aktuell fokussierten Objektes in die Zwischenablage.

- _WinWbnCommandSelCut (2)

 Schneidet die aktuelle Selektion (z. B. in einem Eingabefeld) aus und überträgt ihn in die Zwischenablage.

- _WinWbnCommandSelDelete (31)

 Löscht den selektierten Text oder das Zeichen an der Cursor-Position (z. B. in einem Eingabefeld).

- _WinWbnCommandSelPaste (3)

 Fügt den Inhalt der Zwischenablage in das fokussierte Objekt ein.

- _WinWbnCommandSelSelectAll (4)

 Selektiert den gesamten HTML-Inhalt bzw. den Inhalt des fokussierten Objektes.

- _WinWbnCommandUpdNoCache (10)

 Seite erneut anfordern (entspricht + im Internet Explorer).

- _WinWbnCommandUpdNormal (9)

Kontakt

Seite erneut laden (entspricht  im Internet Explorer).

- `_WinWbnCommandZoomIn (11)`

Seiten-Zoom um 25% vergrößern (entspricht  im Internet Explorer).

- `_WinWbnCommandZoomOut (12)`

Seiten-Zoom um 25% verkleinern (entspricht  im Internet Explorer).

- `_WinWbnCommandZoomReset (13)`

Seiten-Zoom auf 100% setzen (entspricht  im Internet Explorer).



Die Kommandos werden unter folgenden Bedingungen ignoriert:

- Es wird kein HTML-Inhalt angezeigt - Der WebNavigator ist leer oder aber es wird z. B. der Inhalt eines Verzeichnisses angezeigt.
- Das Kommando ist nicht verfügbar - z. B. bei `_WinWbnCommandSelCut / _WinWbnCommandSelDelete`, wenn kein Ausschneiden oder Löschen möglich ist oder bei `_WinWbnCommandNavForward`, wenn noch keine nachfolgende Seite existiert.

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein WebNavigator-Objekt.

In Argument (int1) wurde keines der gültigen Kommandos

ErrValueInvalid angegeben.

with Überprüfung der Objektnamen bei der Übersetzung

Verwandte

Siehe [Befehle](#)

Das **with**-Konstrukt ist ein Sprachelement, welches in Verbindung mit dem Operator **\$:** (nicht zu verwechseln mit **\$**) das Vorhandensein von Oberflächen-Objekten bereits zur Übersetzungszeit sicherstellt.

Wird ein Deskriptor mit dem **\$**-Zeichen referenziert, erfolgt die Überprüfung erst zur Laufzeit der Prozedur. Kann kein Objekt mit dem angegebenen Namen ermittelt werden, erfolgt der Laufzeitfehler "Deskriptor ungültig" und die Prozedur wird abgebrochen.

Mit dem **with**-Konstrukt und dem **\$:-Operator** erfolgt die Überprüfung bereits zum Zeitpunkt der Übersetzung.

Syntax:

```
with <Dialogname> $:<Objektname>>->...
```

Existiert das Objekt oder der Dialog nicht, wird die Übersetzung mit dem Fehler "Name unbekannt" abgebrochen. Als Dialog können folgende Objekte angegeben werden:

- [Frame](#)
- [AppFrame](#)
- [MdiFrame](#)
- [PrintForm](#)
- [PrintFormList](#)
- [PrintDoc](#)
- [PrintDocRecord](#)

Sollen mehrere Anweisungen innerhalb des **with**-Konstrukts verarbeitet werden, müssen diese durch die geschweiften Klammern, in der üblichen Weise geklammert werden.

Beispiele:

```
// Überprüfung eines Objektswith FrmKunden $:edKndNummer->wpCaptionInt # 0;// Überprüfung bei meh
```

with-Konstrukte können geschachtelt werden:

```
with FrmKunden{ ... with FrmSelectionDialog { ... $:rlsSelection->wp... ... } ... }
```

Ist das Objekt **rlsSelection** im Dialog **FrmSelectionDialog** nicht vorhanden, wird dieses im Dialog **FrmKunden** gesucht. Erst wenn es auch dort nicht vorhanden ist, erfolgt ein Übersetzungsfehler. Die Suche erfolgt immer vom inneren zum äußeren **with**-Block.

Mit dem **with**-Konstrukt werden auch die Namensreferenzen auf Dialog-Objekte schneller aufgelöst. Wird während der Übersetzung eine **\$:-Referenz** gefunden, wird der Pfad zu dem entsprechenden Objekt gespeichert. Zur Laufzeit muss nicht mehr

Kontakt

das ganze geladene Objekt durchsucht werden, statt dessen erfolgt die Ermittlung des Deskriptors direkt über den gespeicherten Pfad.

Wurde die Datenbank von einem Stand vor Version 5.0 konvertiert, muss ein Dialog mit der aktuellen Version geöffnet und wieder gespeichert werden, damit die with-Anweisung verwendet werden kann.

Konstanten für Sonstige Befehle

Konstanten für Sonstige Befehle

Siehe **Sonstige Befehle**

- [WinBeepDefault](#)
- [WinBeepError](#)
- [WinBeepInformation](#)
- [WinBeepQuestion](#)
- [WinBeepWarning](#)
- [WinDialogMaximized](#)
- [WinDialogMinimized](#)
- [WinDialogNormal](#)
- [WinDialogPrimary](#)
- [WinFlashCaption](#)
- [WinFlashDefault](#)
- [WinFlashStop](#)
- [WinFlashTray](#)
- [WinFlashUntilFore](#)
- [WinFlashUntilStop](#)
- [WinGntRecalc](#)
- [WinGntRefresh](#)
- [WinImageFormatBmp](#)
- [WinImageFormatJpg](#)
- [WinImageFormatPng](#)
- [WinImageFormatTif](#)
- [WinLayerDarken](#)
- [WinLayerEnd](#)
- [WinLayerStart](#)
- [WinLayerUpdate](#)
- [WinLstActiveOnly](#)
- [WinLstEditClearChanged](#)
- [WinLstEditLst](#)
- [WinLstEditLstAlpha](#)
- [WinLstFromFirst](#)
- [WinLstFromLast](#)
- [WinLstFromSelected](#)
- [WinLstFromTop](#)
- [WinLstIgnoreHidden](#)
- [WinLstPosBottom](#)
- [WinLstPosMiddle](#)
- [WinLstPosSelected](#)
- [WinLstPosTop](#)
- [WinLstRecDoSelect](#)
- [WinLstRecEvtSkip](#)
- [WinLstRecFromBuffer](#)
- [WinLstRecFromRecId](#)
- [WinLstReorderColumns](#)
- [WinMsgAll](#)
- [WinMsgNoInput](#)
- [WinMsgNoKeyboardInput](#)

- [WinMsgNoMouseInput](#)
- [WinPicSignReset](#)
- [WinPicSignSaveAsExt](#)
- [WinPicSignSaveAsJpg](#)
- [WinPicSignSaveAsPng](#)
- [WinPicSignSaveAsTif](#)
- [WinPicSignSavePic](#)
- [WinPicSignSaveSign](#)
- [WinRvwEditAbortEditing](#)
- [WinRvwUpdateDoKeepSelect](#)
- [WinRvwUpdateDoSelect](#)
- [WinRvwUpdateFromFirst](#)
- [WinRvwUpdateFromLast](#)
- [WinRvwUpdateFromRecBuf](#)
- [WinRvwUpdateFromSelected](#)
- [WinRvwUpdateFromTop](#)
- [WinRvwUpdateFromTopLocation](#)
- [WinRvwUpdateOptClearCache](#)
- [WinRvwUpdateOptClearSelected](#)
- [WinUpdActivate](#)
- [WinUpdBuf2Obj](#)
- [WinUpdFld2Buf](#)
- [WinUpdFld2Obj](#)
- [WinUpdObj2Buf](#)
- [WinUpdObj2Fld](#)
- [WinUpdOff](#)
- [WinUpdOn](#)
- [WinUpdSort](#)
- [WinUpdState](#)
- [WinWbnCommandExecPrint](#)
- [WinWbnCommandNavBack](#)
- [WinWbnCommandNavForward](#)
- [WinWbnCommandNavHome](#)
- [WinWbnCommandNavSearch](#)
- [WinWbnCommandSelCopy](#)
- [WinWbnCommandSelCut](#)
- [WinWbnCommandSelDelete](#)
- [WinWbnCommandSelPaste](#)
- [WinWbnCommandSelSelectAll](#)
- [WinWbnCommandUpdNoCache](#)
- [WinWbnCommandUpdNormal](#)
- [WinWbnCommandZoomIn](#)
- [WinWbnCommandZoomOut](#)
- [WinWbnCommandZoomReset](#)

Kontakt

_WinBeepDefault

Systemklang für "Standardton Warnsignal" wiedergeben Wert 0

Siehe WinBeep()

Option bei WinBeep() durch die der Systemklang für "Standardton Warnsignal" wiedergegeben werden kann.

Kontakt

_WinBeepError

Systemklang für "Kritischer Fehler" wiedergeben Wert 3

Siehe WinBeep()

Option bei WinBeep() durch die der Systemklang für "Kritischer Fehler"
wiedergegeben werden kann.

Kontakt

_WinBeepInformation

Systemklang für "Sternchen" wiedergeben

Wert 2

Siehe WinBeep()

Option bei WinBeep() durch die der Systemklang für "Sternchen" wiedergegeben werden kann.

Kontakt

_WinBeepQuestion

Systemklang für "Frage" wiedergeben

Wert 5

Siehe WinBeep()

Option bei WinBeep() durch die der Systemklang für "Frage" wiedergegeben werden kann.

Kontakt

_WinBeepWarning

Systemklang für "Hinweis" wiedergeben

Wert 4

Siehe [WinBeep\(\)](#)

Option bei [WinBeep\(\)](#) durch die der Systemklang für "Hinweis" wiedergegeben werden kann.

Kontakt

_WinDialogMaximized

Fenster wird maximiert dargestellt

Wert 16 / 0x00000010

Verwandte

Befehle,

WinDialog(),

Siehe WinDialogRun(),

WinAdd(),

WinAddByName(),

WinUpdate(),

WinInfo()

Je nach Befehl hat _WinDialogMaximized folgende Bedeutung:

- WinDialog(), WinDialogRun(), WinAdd() und WinAddByName() Dialog

wird maximiert geladen.

- WinUpdate()

Dialog wird zur Laufzeit maximiert. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.

- WinInfo()

Liefert WinInfo(_WinState) den Wert _WinDialogMaximized zurück, ist der Dialog maximiert.

Kontakt

_WinDialogMinimized
Fenster wird minimiert dargestellt
Wert 32 / 0x00000020

Verwandte

Befehle,

WinDialog(),

Siehe WinDialogRun(),

WinAdd(),

WinAddByName(),

WinUpdate(),

WinInfo()

Je nach Befehl hat **_WinDialogMinimized** folgende Bedeutung:

- WinDialog(), WinDialogRun(), WinAdd() und WinAddByName() Dialog wird minimiert geladen.
- WinUpdate()
Dialog wird zur Laufzeit minimiert. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.
- WinInfo()
Liefert WinInfo(_WinState) den Wert **_WinDialogMinimized** zurück ist der Dialog minimiert.

_WinDialogNormal

Fenster weder maximiert noch minimiert
Wert 64 /

0x00000040

Siehe WinUpdate(), WinInfo(),

Je nach Befehl hat _WinDialogNormal folgende Bedeutung:

- WinDialog() und WinDialogRun()

Der Dialog wird weder minimiert, noch maximiert angezeigt. Ist durch die Darstellung das Fenster in keinem Bildschirm sichtbar, wird es im primären Bildschirm zentriert dargestellt.

- WinUpdate()

Dialog wird zur Laufzeit von einem minimierten oder maximierten Status zurückgesetzt. Die Option kann nur angegeben werden, wenn WinUpdState als ersten Parameter übergeben wird.

- WinInfo()

Liefert WinInfo(_WinState) den Wert _WinDialogNormal zurück ist der Dialog weder maximiert noch minimiert.

Kontakt

_WinDialogPrimary

Fenster wird auf dem Primär-Bildschirm zentriert

Wert 128 /

0x00000080

Verwandte

Siehe Befehle,

WinUpdate()

Ist diese Konstante bei WinUpdate() angegeben und befindet sich das AppFrame- oder Frame-Objekt in einem nicht angezeigten Bereich, wird das Fenster auf der Bildschirmmitte des Primärbildschirms dargestellt.

Die Anweisung wird ignoriert, wenn sich das Fenster (auch nur teilweise) im darstellbaren Bereich eines Bildschirms befindet.

Kontakt

_WinFlashCaption

Titelzeile blinken lassen

Wert 1 / 0x0001

[WinFlash\(\)](#),

Siehe [WinFlashTray](#),

[WinFlashDefault](#)

Option bei [WinFlash\(\)](#), mit der die Titelzeile des angegebenen Fensters blinkt.

Kontakt

_WinFlashDefault

Titelzeile und Taskleisten-Icon blinken lassen Wert 3

/ 0x0003

WinFlash(),

Siehe WinFlashCaption,

WinFlashTray

Option bei WinFlash(), mit der die Titelzeile und das Icon in der Taskleise des angegebenen Fensters blinken.

Kontakt

_WinFlashStop

Blinken des Fensters stoppen

Wert 0 / 0x0000

WinFlash(),

Siehe WinFlashUntilStop,

WinFlashUntilFore

Option bei WinFlash(), mit der das Blinken des Fenster gestoppt wird, unabhängig davon mit welcher Option WinFlash() initiiert wurde.

Kontakt

_WinFlashTray

Taskleisten-Icon blinken lassen

Wert 2 / 0x0002

[WinFlash\(\)](#),

Siehe [WinFlashCaption](#),

[WinFlashDefault](#)

Option bei [WinFlash\(\)](#), mit der das Icon in der Taskleise des angegebenen Fensters blinks.

Kontakt

_WinFlashUntilFore

Fenster blinken lassen, bis es in den Vordergrund geholt wird Wert 12 / 0x000C

WinFlash(),

Siehe WinFlashUntilStop,

WinFlashStop

Option bei WinFlash(), mit der das Fenster so lange blinkt, bis es in den Vordergrund geholt wird.

Kontakt

_WinFlashUntilStop

Fenster blinken lassen, bis WinFlash() mit WinFlashStop aufgerufen oder das Fenster beendet wird

Wert 4 / 0x0004

WinFlash(),

Siehe WinFlashStop,

WinFlashUntilFore

Option bei WinFlash(), mit der das Fenster so lange blinkt, bis WinFlash() mit der Option WinFlashStop aufgerufen oder das Fenster beendet wird.

Kontakt

_WinGntRecalc

GanttGraph-Objekt neu berechnen

Wert 1 / 0x00000001

WinUpdate(),

Siehe WinUpdState,

WinGntRefresh

Mit der Anweisung `$GanttGraph->WinUpdate(WinUpdState, WinGntRecalc)` wird das GanttGraph-Objekt neu berechnet und anschließend die Achsen und die View-Bereiche neu gezeichnet.

Eine neue Berechnung des GanttGraph-Objektes wird notwendig, wenn eine der folgenden Eigenschaften der Achse geändert wurden:

- Size
- SizeScala
- TitleText

Die Aktualisierung mit der Option _WinGntRecalc schließt die Aktualisierung mit der Option WinGntRefresh mit ein.

Die Anweisung wird nicht durchgeführt, wenn die Eigenschaft AutoUpdate auf false gesetzt ist.

Kontakt

_WinGntRefresh

GanttGraph-Objekt neu zeichnen

Wert 0 / 0x00000000

WinUpdate(),

Siehe WinUpdState,

WinGntRecalc

Mit der Anweisung `$GanttGraph->WinUpdate(WinUpdState, WinGntRefresh)` wird das GanttGraph-Objekt neu gezeichnet.

Eine Aktualisierung des Objektes wird notwendig, wenn eine Eigenschaft einer Achse verändert wurde. Findet eine Änderung der Größe der Achse statt, muss der Parameter WinGntRecalc verwendet werden. Die Größe wird beim Setzen der Eigenschaften Size, SizeScala und TitleText verändert.

Die Anweisung wird nicht durchgeführt, wenn die Eigenschaft AutoUpdate auf false gesetzt ist.

Kontakt

_WinImageFormatBmp

Windows-Bitmap-Datei erstellen

Wert 2

WinBarcodeSaveImage(),

Siehe WinImageFormatJpg,

WinImageFormatPng,

WinImageFormatTif

Option bei WinBarcodeSaveImage(), durch die der Inhalt eines Barcode-Objektes als Windows-Bitmap-Datei gespeichert wird.

Kontakt

_WinImageFormatJpg

JPEG-Datei erstellen

Wert 3

WinBarcodeSaveImage(),

Siehe WinImageFormatBmp,

WinImageFormatPng,

WinImageFormatTif

Option bei WinBarcodeSaveImage(), durch die der Inhalt eines Barcode-Objektes als JPEG-Datei gespeichert wird.

Kontakt

_WinImageFormatPng

PNG-Datei erstellen

Wert 5

WinBarcodeSaveImage(),

Siehe WinImageFormatBmp,

WinImageFormatJpg,

WinImageFormatTif

Option bei WinBarcodeSaveImage(), durch die der Inhalt eines Barcode-Objektes als PNG-Datei gespeichert wird.

Kontakt

_WinImageFormatTif

TIFF-Datei erstellen

Wert 4

WinBarcodeSaveImage(),

Siehe WinImageFormatBmp,

WinImageFormatJpg,

WinImageFormatPng

Option bei WinBarcodeSaveImage(), durch die der Inhalt eines Barcode-Objektes als TIFF-Datei gespeichert wird.

Kontakt

_WinLayerDarken

Fenster abdunkeln

Wert 1 / 0x00000001

Verwandte

Siehe [Befehle](#),

[WinLayer\(\)](#),

[WinLayerDCross](#)

Option bei [WinLayer\(\)](#) um beim Layer-Modus eines Fensters den Anwendungsbildschirm etwas abzudunkeln.

Kontakt

_WinLayerDCross

Fenster mit diagonaler Kreuz-Schraffur überdecken Wert 2 /
0x00000002

Verwandte

Siehe Befehle,

WinLayer(),

WinLayerDarken

Option bei WinLayer() um beim Layer-Modus das Fenster mit einer diagonalen Kreuz-Schraffur zu überdecken.

Kontakt

_WinLayerEnd

Layer-Modus für Fenster beenden

Wert 1 /

0x00000001

Verwandte

Siehe Befehle,

WinLayer()

Option bei WinLayer() um den Layer-Modus eines Fensters zu beenden.

Kontakt

_WinLayerStart

Layer-Modus für Fenster aktivieren
Wert 0 /

0x00000000

Verwandte

Siehe Befehle,

WinLayer()

Option bei WinLayer() um den Layer-Modus eines Fensters zu aktivieren.

Kontakt

_WinLayerUpdate

Fenster im Layer-Modus aktualisieren
Wert 2 /

0x00000002

Verwandte

Siehe Befehle,

WinLayer()

Option bei WinLayer() um ein Fenster im Layer-Modus zu aktualisieren.

Kontakt

_WinLstActiveOnly

Nur das aktive View aktualisieren

-2.147.483.648

Wert / 0x80000000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Aktualisiert bei RecList- und DataList-Objekten mit mehreren Views nur das aktive View.

Kontakt

_WinLstEditLst

Liste mit einer Spalte im Eingabe-Objekt erzeugen Wert 1 /
0x00000001

Verwandte

Siehe Befehle,

WinLstEdit(),

EvtLstEditStart

Option bei WinLstEdit() durch die im Eingabe-Objekt der Liste eine Liste mit einer Spalte erzeugt werden kann.

Die Liste kann im Ereignis EvtLstEditStart mit Einträgen gefüllt werden.

Kontakt

_WinLstEditClearChanged

Änderungs-Flag zurücksetzen

Wert 16.777.216 /
0x01000000

Vewandte

Siehe Befehle,

WinLstEdit()

Option bei WinLstEdit() durch die das Änderungs-Flag zurückgesetzt werden kann.

Das Setzen des Flags erfolgt nach dem Ausführen des Ereignisses EvtLstEditCommit, wenn dieses Ereignis den Wert true zurückgibt. Der Inhalt des Flags wird im Ereignis EvtLstEditFinished übergeben und kann dort ausgewertet werden.

Kontakt

_WinLstEditLstAlpha

Liste mit zwei Spalten in Eingabe-Objekt erzeugen Wert 2 /
0x00000002

Verwandte

Siehe Befehle,

WinLstEdit(),

EvtLstEditStart

Option bei WinLstEdit() durch das im Eingabe-Objekt der Liste eine Liste mit zwei Spalten erzeugt werden kann.

Die Liste kann im Ereignis EvtLstEditStart mit Einträgen gefüllt werden.

Kontakt

_WinLstFromFirst

Ab dem ersten Datensatz

Wert 1 /

0x00000001

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Liest ab dem ersten Datensatz der Datei die Datensätze der RecList neu ein. Es werden nur soviele Sätze gelesen, wie in der Liste dargestellt werden.

Kontakt

_WinLstFromLast

Ab dem letzten Datensatz

Wert 2 /

0x00000002

Verwandte

Siehe **Befehle**,

WinUpdate()

Option bei **WinUpdate()**. Liest ab dem letzten Datensatz der Datei die Datensätze der **RecList** neu ein. Es werden nur soviele Sätze gelesen, wie in der Liste dargestellt werden.

Kontakt

_WinLstFromSelected

Ausgehend vom selektierten Datensatz
Wert ^{32 /}

0x00000020

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Der Datensatz, der den Fokus hat, bestimmt die Datensätze, die angezeigt werden. Über die Option _WinLstPos... wird die Position angegeben, die der Datensatz in der Liste einnimmt.

Kontakt

_WinLstFromTop

Ab dem ersten sichtbaren Datensatz
Wert ^{64 /}

0x00000040

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Der erste angezeigte Datensatz in der RecList bestimmt die Datensätze, die angezeigt werden. Über die Option _WinLstPos... wird die Position angegeben, die der Datensatz einnimmt.

Kontakt

_WinLstIgnoreHidden

Update, auch wenn die Liste nicht sichtbar ist Wert

1.073.741.824

/ 0x40000000

Verwandte

Siehe **Befehle**,

WinUpdate()

Option bei **WinUpdate()**. Das Update wird auch durchgeführt, wenn die Liste oder eines der Elternobjekte nicht sichtbar (**Visible = false**) ist. In der Regel muss diese Option nicht angegeben werden.

Kontakt

_WinLstPosBottom

Letzter sichtbarer Satz in der RecList

Wert 16.384 /
0x00004000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Der in den Feldpuffern befindliche Datensatz erscheint als letzter in der RecList.

Kontakt

_WinLstPosMiddle

Mittelposition in der RecList

Wert 8.192 /

0x00002000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Der in den Feldpuffern befindliche Datensatz erscheint in der Mitte der RecList.

Kontakt

_WinLstPosSelected

Datensatz behält die Position innerhalb der Liste

Wert 32.768 /

0x00008000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Die Option bewirkt einen Neuaufbau der Liste, wobei die Position des selektierten Satzes innerhalb der Liste erhalten bleibt.

_WinLstPosSelected ist in Kombination mit _WinLstFromSelected und _WinLstRecDoSelect zu verwenden.

Beispiel:

```
$RecList->WinUpdate(_WinUpdOn, _WinLstFromSelected | _WinLstPosSelected | _WinLstRecDoSelect);
```

Kontakt

_WinLstPosTop

Erster sichtbarer Satz in der RecList

Wert 4.096 /

0x00001000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Der in den Feldpuffern befindliche Datensatz erscheint als erster in der RecList.

Kontakt

_WinLstRecDoSelect

Datensatz selektieren

256 /

Wert

0x00000100

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). In der RecList bekommt der Datensatz den Fokus, der in den Feldpuffern eingetragen ist. Die Position wird über die Option _WinLstPos... bestimmt.

Kontakt

_WinLstRecEvtSkip

Unterbinden des Ereignisses EvtLstSelect

Wert 131.072 /

0x00020000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Mit dieser Option wird das Auslösen des Ereignisses EvtLstSelect bei einem Update auf die RecList unterbunden.

Kontakt

_WinLstRecFromBuffer

Ausgehend von den Werten des Feldpuffers
Wert¹⁶/

0x00000010

Verwandte

Siehe **Befehle**,

WinUpdate()

Option bei **WinUpdate()**. Der Inhalt der Feldpuffer bestimmt die Datensätze, die angezeigt werden.
Über die Option **_WinLstPos...** wird die Position angegeben, die der Datensatz einnimmt.

Existiert der Datensatz der in dem Feldpuffer steht nicht mehr, oder kann er durch einen gesetzten Filter nicht mehr angezeigt werden, wird der nächste darstellbare Datensatz an der entsprechenden Position angezeigt.

Kontakt

_WinLstRecFromRecId

Ausgehend von der Datensatz-ID
Wert 128 /

0x00000080

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Die Datensatz-ID bestimmt, von welchem Satz aus die Liste aufgebaut wird. Über die Option _WinLstPos... wird die Position angegeben, die der Datensatz einnimmt.

Kontakt

_WinLstReorderColumns

Spalten neu nummerieren

Wert 268.435.456 /

0x10000000

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Alle sichtbaren Spalten einer Liste werden neu nummeriert. Anschließend haben alle sichtbaren Spalten eine fortlaufende ClmOrder beginnend mit 1.

Kontakt

_WinMsgAll

Alle Windows Nachrichten verarbeiten

Wert 0

WinSleep(),

Siehe — WinMsgNoKeyboardInput WinMsgNoMouseInput

WinMsgNoInput

Wird diese Konstante bei WinSleep() oder keine der anderen _WinMsg...--Konstanten angegeben, werden alle Windows-Nachrichten aus der Nachrichtenschleife verarbeitet. Führt die Verarbeitung einer Nachricht zur Ausführung eines Ereignisses, welches bereits ausgeführt wird, wird die Nachricht verworfen.

Kontakt

_WinMsgNoInput

Keine Tastatur- oder Mausnachrichten verarbeiten Wert 3

WinSleep(), WinMsgAll

Siehe WinMsgNoKeyboardInput

WinMsgNoMouseInput

Wird diese Konstante bei WinSleep() angegeben, werden keine Tastatur- oder Mausnachrichten aus der Nachrichtenschleife verarbeitet. Sie verbleiben in der Nachrichtenschleife und werden nach dem WinSleep() ausgeführt.

Kontakt

_WinMsgNoKeyboardInput

Keine Tastaturnachrichten verarbeiten

Wert 1

[WinSleep\(\)](#), [WinMsgAll](#)

Siehe [WinMsgNoMouseInput](#)

[WinMsgNoInput](#)

Wird diese Konstante bei [WinSleep\(\)](#) angegeben, werden keine Tastaturnachrichten aus der Nachrichtenschleife verarbeitet. Sie verbleiben in der Nachrichtenschleife und werden nach dem [WinSleep\(\)](#) ausgeführt.

Kontakt

_WinMsgNoMouseInput

Keine Mausnachrichten verarbeiten

Wert 2

WinSleep(), WinMsgAll

Siehe WinMsgNoKeyboardInput

WinMsgNoInput

Wird diese Konstante bei WinSleep() angegeben, werden keine Mausnachrichten aus der Nachrichtenschleife verarbeitet. Sie verbleiben in der Nachrichtenschleife und werden nach dem WinSleep() ausgeführt.

Kontakt

_WinPicSignSaveAsExt

Speicherformat aus Dateiendung ermitteln

Wert 1 / 0x0001

Siehe WinPicSaveImage()

Option bei WinPicSaveImage(), durch die das Dateiformat beim Speichern aus der Dateiendung ermittelt wird.

Kontakt

_WinPicSignSaveAsJpg

Zeichnung im JPEG-Format speichern

Wert 2 / 0x0002

Siehe WinPicSaveImage()

Option bei WinPicSaveImage(), durch die die Zeichnung im JPEG-Format gespeichert wird.

Kontakt

_WinPicSignSaveAsPng

Zeichnung im PNG-Format speichern

Wert 4 / 0x0004

Siehe WinPicSaveImage()

Option bei WinPicSaveImage(), durch die die Zeichnung im PNG-Format gespeichert wird.

Kontakt

_WinPicSignSaveAsTif

Zeichnung im TIFF-Format speichern

Wert 8 / 0x0008

Siehe [WinPicSaveImage\(\)](#)

Option bei [WinPicSaveImage\(\)](#), durch die die Zeichnung im TIFF-Format gespeichert wird.

Kontakt

_WinPicSignSavePic

Inhalt des Picture-Objektes speichern

Wert 256 / 0x0100

Siehe WinPicSaveImage()

Option bei WinPicSaveImage(), durch die der Inhalt des Picture-Objektes gespeichert wird.

Kontakt

_WinPicSignReset

Zeichnung zurücksetzen

Wert 1.024 / 0x0400

Siehe WinPicSaveImage()

Option bei WinPicSaveImage(), durch die die Zeichnung zurückgesetzt und nur das originale Bild im Picture-Objekt angezeigt wird.

Kontakt

_WinPicSignSaveSign

Zeichnung speichern

Wert 512 / 0x0200

Siehe WinPicSaveImage()

Option bei WinPicSaveImage(), durch die die Zeichnung gespeichert wird.

Kontakt

_WinRvwEditAbortEditing

aktiven Editievorgang abbrechen

Wert 1 / 0x01

Siehe [WinRvwEdit\(\)](#)

Bei dem Befehl [WinRvwEdit\(\)](#) kann diese Konstante im Parameter (int4) angegeben werden, um einen aktiven Editievorgang abzubrechen, bevor das ausgewählte Item oder SubItem bearbeitet wird.

Kontakt

_WinRvwUpdateDoKeepSelect

Selektierten Datensatz weiterhin selektieren Wert

512 / 0x00000200

Verwandte

Siehe Befehle,

WinRvwUpdate()

Option bei WinRvwUpdate(). In dem RecView behält der aktuell selektierte Datensatz den Fokus.



Pro RecView-Objekt gibt es nur einen selektierten Datensatz.

Kontakt

_WinRvwUpdateDoSelect

Aktuellen Datensatz aus den Feldpuffern selektieren
Wert 256 /

0x00000100

Verwandte

Siehe Befehle,

WinRvwUpdate(),

EvtLstViewInit

Option bei WinRvwUpdate(). In dem RecView bekommt der Datensatz den Fokus, der in den Feldpuffern eingetragen ist.



Pro RecView-Objekt gibt es nur einen selektierten Datensatz.

Kontakt

_WinRvwUpdateFromFirst

Ab dem ersten Datensatz

Wert 1 / 0x00000001

Verwandte

Siehe Befehle,

WinRvwUpdate(),

EvtLstViewInit

Option bei WinRvwUpdate(). Liest ab dem ersten Datensatz der Datei die Datensätze des RecView neu ein. Es werden nur soviele Sätze gelesen, wie in dem View dargestellt werden.

Kontakt

_WinRvwUpdateFromLast

Ab dem letzten Datensatz

Wert 2 / 0x00000002

Verwandte

Siehe Befehle,

WinRvwUpdate(),

EvtLstViewInit

Option bei WinRvwUpdate(). Liest ab dem letzten Datensatz der Datei die Datensätze des RecView neu ein. Es werden nur soviele Sätze gelesen, wie in dem View dargestellt werden.

Kontakt

[_WinRvwUpdateFromRecBuf](#)

Ausgehend von den Werten des Feldpuffers.

Wert 5 / 0x00000005

Verwandte

Siehe [Befehle](#),

[WinRvwUpdate\(\)](#),

[EvtLstViewInit](#)

Option bei [WinRvwUpdate\(\)](#). Der Inhalt der Feldpuffer bestimmt die Datensätze, die angezeigt werden.

Existiert der Datensatz der in dem Feldpuffer steht nicht mehr, oder kann er durch einen gesetzten Filter nicht mehr angezeigt werden, wird der nächste darstellbare Datensatz angezeigt.

Kontakt

_WinRvwUpdateFromSelected

Ab dem ersten sichtbaren Datensatz

Wert 4 / 0x00000004

Verwandte

Siehe Befehle,

WinRvwUpdate()

Option bei WinRvwUpdate(). Der erste angezeigte Datensatz in dem RecView ist die aktuell selektierte Gruppe.

Kontakt

_WinRvwUpdateFromTop

Ab dem ersten sichtbaren Datensatz

Wert 3 / 0x00000003

Verwandte

Siehe Befehle,

WinRvwUpdate(),

EvtLstViewInit

Option bei WinRvwUpdate(). Der erste angezeigte Datensatz in dem RecView bestimmt die Datensätze, die angezeigt werden.

Kontakt

_WinRvwUpdateFromTopLocation

Angezeigte Datensätze aktualisieren und Position der ersten angezeigten Gruppe beibehalten

Wert 6 / 0x00000006

Verwandte

Siehe Befehle,

WinRvwUpdate(),

EvtLstViewInit

Option bei WinRvwUpdate(). Der Befehl mit dieser Option führt einen Neuaufbau der angezeigten Datensätze des RecView-Objektes durch, ohne die Position der ersten angezeigten Gruppe zu ändern.

Kontakt

_WinRvwUpdateOptClearCache

Cache des RecView-Objektes leeren

Wert 65.536 / 0x00010000

Verwandte Befehle,

Siehe WinRvwUpdate(),

WinRvwUpdateOptClearSelected

Option bei WinRvwUpdate(). Alle Gruppen werden aus dem Cache des RecView-Objektes entfernt.

Dies hat zur Folge, dass das Ereignis EvtLstGroupInit für alle anzuzeigenden Gruppen durchgeführt wird.

Kontakt

_WinRvwUpdateOptClearSelected

Wert 524.288 / 0x00080000

Verwandte Befehle,

Siehe [WinRvwUpdate\(\)](#),

[WinRvwUpdateOptClearCache](#)

Option bei [WinRvwUpdate\(\)](#). Die aktuell selektierte Gruppe wird aus dem Cache des RecView-Objektes entfernt. Dies hat zur Folge, dass das Ereignis EvtLstGroupInit für diese Gruppe durchgeführt wird.

Kontakt

_WinUpdActivate

Frame-Objekt in den Vordergrund holen

Wert 30

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Holt das Fenster-Objekt in den Vordergrund.



Die Aktivierung lässt sich nicht erzwingen. Liegt eine andere Anwendung über dem Fenster der CONZEPT 16-Applikation, verhindert Windows, dass diese Anwendung den Fokus verliert. Dies führt zum Blinken des Fensters, welches aktiviert werden soll.

Kontakt

_WinUpdBuf2Obj

Übertragen des Textpuffers

Wert 13

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Überträgt den Inhalt des Textpuffers in das Objekt TextEdit.

Kontakt

_WinUpdFld2Buf

Übertragen der Feldpuffer in den Datensatzpuffer Wert 11

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate() - Überträgt den Feldpuffer in den Datensatzpuffer der Objekte RecList und MdiFrame.

- RecList

Ist die Eigenschaft DbRecBuf einer RecList gesetzt, werden Änderungen der Feldpuffer die durch die RecList verursacht werden, auf den ursprünglichen Feldpufferinhalt zurückgesetzt. Dies ist auch der Fall, wenn in dem Ereignis EvtLstSelect das Lesen eines Datensatzes stattfindet. Soll der Feldpuffer des in diesem Ereignis gelesenen Datensatzes erhalten bleiben, muss der Feldpuffer in den Puffer der RecList übertragen werden.

Beispiel:

```
sub EvtLstSelect( aEvt    : event;      // Ereignis  aID      : bigint;    // Datensatz-ID) : 1
```

- MdiFrame

Bei einem MdiFrame überträgt der Befehl die Inhalte der Feldpuffer in den Datensatzpuffer (DbRecBuf) des Objektes.

Beispiel:

```
$MdiFrame->WinUpdate(_WinUpdFld2Buf);
```

Wird der Befehl mit einem Objekt benutzt, welches kein Fenster-Objekt und kein RecList-Objekt ist, wird das Fenster-Objekt selbstständig ermittelt.

Beispiel:

```
$button->WinUpdate(_WinUpdFld2Buf);
```

Es wird zunächst das übergeordnete Fenster-Objekt des Buttons ermittelt. Dann werden die Feldpuffer in die Datensatzpuffer des Fenster-Objektes übertragen.

Kontakt

_WinUpdFld2Obj

Übertragen der Feldpuffer

Wert 10

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Überträgt den Inhalt der Feldpuffer in das Objekt, sofern in der Eigenschaft DbFieldName des Objektes ein Datenbankfeld angegeben ist.

Kontakt

_WinUpdObj2Buf

Übertragen des Inhaltes des Objektes in den Textpuffer Wert 12

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Überträgt den Inhalt des Objektes TextEdit in den Textpuffer.

Kontakt

_WinUpdObj2Fld

Übertragen der Objektinhalte

Wert 14

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Überträgt den Inhalt des Eingabeobjektes in den Feldpuffer, sofern in der Eigenschaft DbFieldName des Objektes ein Datenbankfeld angegeben ist.

_WinUpdOff

Eigenschaft AutoUpdate auf false setzen

Wert 21

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Setzt die Eigenschaft AutoUpdate eines Objektes auf false. Der Parameter kann mit der Option WinUpdScrollPos kombiniert werden, um bei Objekten mit einem Scrollbar die Scrollposition zu speichern. Die Scrollposition muss kann später mit der Kombination WinUpdOn | WinUpdScrollPos wieder hergestellt werden.

Kontakt

_WinUpdOn

Eigenschaft AutoUpdate auf true setzen

Wert 20

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Setzt die Eigenschaft AutoUpdate eines Objektes auf true. Der Parameter kann mit der Option WinUpdScrollPos kombiniert werden, um bei Objekten mit einem Scrollbar die Scrollposition wieder herzustellen. Die Scrollposition muss zuvor mit der Kombination WinUpdOff | WinUpdScrollPos gesichert worden sein.

Kontakt

_WinUpdScrollPos

Speichern und Wiederherstellen der Scrollposition Wert

2.147.483.648 0x80000000

Verwandte

Siehe Befehle,

WinUpdate()

Bei der Anweisung WinUpdate() können die Konstanten WinUpdOff und WinUpdOn mit dieser Konstante kombiniert werden.

Die Anweisung tObj->WinUpdate(_WinUpdOff | _WinUpdScrollPos) speichert die derzeitige Scrollposition und setzt die Eigenschaft AutoUpdate auf false. Zu einem späteren Zeitpunkt kann mit der Anweisung tObj->WinUpdate(_WinUpdOn | _WinUpdScrollPos) die ursprüngliche Scrollposition wieder hergestellt werden.

Die Kombination kann bei allen Objekten verwendet werden, die eine normale Scrollbar (Listen-Objekte, TextEdit usw.) besitzen. Besitzt das Objekt keine normale Scrollbar, zum Beispiel bei einem RecList-Objekt mit dem Ereignis EvtLstRecControl, kann die Scrollposition nicht gesichert werden.

Wird die Scrollposition wieder hergestellt, ohne die Position zuvor zu sichern, erfolgt der Laufzeitfehler ErrValueInvalid.

Beispiele:

```
aEvt:Obj->WinUpdate(_WinUpdOff | _WinUpdScrollPos);...aEvt:Obj->WinUpdate(_WinUpdOn | _WinUpdScro
```

Kontakt

_WinUpdSort

Sortierung aktualisieren

Wert 32

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate(). Der Inhalt eines DataList-Objekts wird neu sortiert. Die Anweisung muss aufgerufen werden, wenn Änderungen am Inhalt von einer oder mehreren Zeilen vorgenommen wurden.

Kontakt

_WinUpdState

Status setzen

Wert 31

Verwandte

Siehe Befehle,

WinUpdate()

Option bei WinUpdate().

Bei Angabe dieser Option muss in (int2) eine der folgenden Konstanten verwendet werden:

WinDialogMaximized

Dialog maximieren

WinDialogMinimized

Dialog minimieren

WinDialogNormal

Dialog weder maximieren noch minimieren

WinGntRefresh

GanttGraph-Objekt neu zeichnen

WinGntRecalc

GanttGraph-Objekt neu berechnen

WinStateAttachClosed

GroupTile-Objekt unsichtbar setzen

WinStateAttachMaximized GroupTile-Objekt maximieren. Ist das GroupTile-Objekt vor der Ausführung unsichtbar, wird es automatisch auf sichtbar gesetzt.

WinStateAttachNormal GroupTile-Objekt normal darstellen. Ist das GroupTile vor der Ausführung unsichtbar, wird es automatisch auf sichtbar gesetzt.

Beispiele:

```
// Dialog maximieren$Frame->WinUpdate(_WinUpdState, _WinDialogMaximized); // Dialog minimieren$Fra
```

Kontakt

Befehle der Mehrfachselektion

Befehle zur Programmierung einer Mehrfachselektion Siehe
Objekte der Mehrfachselektion,
Mehrfachselektions-Ereignisse

Befehle:

- WinMsdDelete
- WinMsdDeleteName
- WinMsdInsert
- WinMsdInsertName
- WinMsdRead
- WinMsdReadName
- WinMsdUpdate

Konstanten:

- WinMsdNoTreeSync
- WinMsdRecId

Kontakt

obj -> WinMsdDelete(bigint1[, int2]) : int



Element aus der Selektion entfernen

Deskriptor des Objekts, in dem selektiert
obj wird

bigint1 Element, das aus der Selektion entfernt werden soll

Optionen (optional)

WinMsdNoTreeSync TreeView-Objekt

int2 nicht neu aufbauen

WinMsdRecId Datensatz-ID

entfernen

Fehlerwert

Element entfernt.

Resultat **int** ErrOk Das Element wurde nicht gefunden.

ErrMsdNotFound Das Element wurde nicht gefunden.

Verwandte Befehle, WinMsdInsert(),

Siehe WinMsdRead(), WinMsdDeleteName()

Mit diesem Befehl wird das Element (bigint1) aus der Menge der selektierten Elemente entfernt.

Das Objekt, in dem die Selektion stattfindet wird als (obj) der Anweisung übergeben. In (bigint1) wird das Element übergeben, dass aus der Selektion entfernt werden soll. In Abhängigkeit vom Objekt muss in (bigint1) eine Datensatz-ID (RecList), eine Zeilennummer (DataList), der Deskriptor eines TreeNode-Objekts (TreeView), der Deskriptor eines CanvasGraphic-Objekts (Canvas) oder der Deskriptor eines CteItems (SelectionData) übergeben werden.

Beim RecList-Objekt kann auch die Option WinMsdRecId verwendet werden. In (bigint1) muss dann die Dateinummer angegeben werden. Dadurch wird automatisch die Datensatz-ID des aktuell geladenen Datensatzes der angegebenen Datei ermittelt.

Bei den Objekten RecList und DataList wird das neu hinzugefügte Selektions-Element sichtbar, nachdem das Objekt durch den Befehl WinUpdate() neu gezeichnet wurde. Beim Canvas-Objekt muss zusätzlich die Option WinUpdState angegeben werden. Beim TreeView-Objekt müssen die Selektions-Elemente explizit übertragen werden, damit die Selektion im TreeView-Objekt sichtbar wird. Bei Verwendung von WinMsdDelete() ist dies standardmäßig der Fall. Durch WinMsdNoTreeSync kann die Übertragung des neuen Elementes in den TreeView verhindert werden. Um alle Elemente einer Mehrfachselektion in das TreeView-Objekt zu übertragen kann der Befehl WinMsdUpdate() benutzt werden.

Natürlich kann zum Entfernen eines Selektions-Elementes auch der Befehl CteDelete() verwendet werden. Beim TreeView-Objekt wird anschließend eine Übertragung der Elemente in der Selektionsmenge in das TreeView-Objekt notwendig. Hierzu kann der Befehl WinMsdUpdate() verwendet werden.

Beispiele:

```
// Entfernen eines Datensatzes$RecList->WinMsdDelete(tblCstCustomer, _WinMsdRecId); // Identisch m
```

Kontakt

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Die in (int2) übergebenen Optionen sind nicht zulässig.

ErrFileInvalid Die angegebene Dateinummer in (int1) existiert nicht.

Kontakt

obj -> WinMsdDeleteName(alpha1) : handle Element aus der Selektion einer StoList entfernen



obj Deskriptor des Objekts, in dem selektiert wird

alpha1 Element, das aus der Selektion entfernt werden soll

Fehlerwert

ErrOk

Element entfernt.

Resultat int

ErrMsdNotFound Das Element wurde nicht gefunden.

Siehe

Verwandte Befehle, WinMsdInsertName(),

WinMsdReadName(), WinMsdDelete()

Mit diesem Befehl wird das Element (alpha1) aus der Menge der selektierten Elemente entfernt.

Das Objekt, in dem die Selektion stattfindet wird als (obj) der Anweisung übergeben.

In (alpha1) wird das Element übergeben, dass aus der Selektion entfernt werden soll.

Natürlich kann zum Entfernen eines Selektions-Elementes auch der Befehl CteDelete() verwendet werden.



Dieser Befehl kann nur mit StoList-Objekten verwendet werden.

Beispiele:

```
$StoList->WinMsdDeleteName('StoDir');
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Die in (alpha2) übergebene Zeichenkette ist leer.

Kontakt

obj ->

WinMsdInsert(bigint1[, int2[, alpha3]]) : int

Selektion erweitern

Deskriptor des Objekts, in
obj

dem selektiert wird

bigint1 Element, das in die Selektion

aufgenommen werden soll

Optionen (optional)

WinMsdNoTreeSync TreeView-Objekt

nicht neu

int2

aufbauen

WinMsdRecId

Datensatz-ID
einfügen

alpha3 Custom-Eigenschaft

Fehlerwert

ErrOk

Element
aufgenommen.

Resultat int

ErrMsdExists Das Element
ist bereits

Verwandte Befehle,
WinMsdDelete(),

Siehe

WinMsdRead(),
WinMsdInsertName()

Mit diesem Befehl wird das Element (bigint1) in die Menge der selektierten Elemente hinzugefügt.

Das Objekt, in dem die Selektion stattfindet wird als (obj) der Anweisung übergeben. In (bigint1) wird das Element übergeben, das in die Selektion aufgenommen werden soll. In Abhängigkeit vom Objekt muss in (bigint1) eine Datensatz-ID (RecList), eine Zeilenummer (DataList), der Deskriptor eines TreeNode-Objekts (TreeView), der Deskriptor eines CanvasGraphic-Objektes (Canvas) oder der Deskriptor eines Cte-Items (SelectionData) übergeben werden.

Beim RecList-Objekt kann auch die Option WinMsdRecId verwendet werden. In (bigint1) muss dann die Dateinummer angegeben werden. Dadurch wird automatisch die Datensatz-ID des aktuell geladenen Datensatzes der angegebenen Datei ermittelt.

In (alpha3) kann eine Zeichenkette angegeben werden, die in der Custom-Eigenschaft des Cte-Items gespeichert wird.

Bei den Objekten RecList und DataList wird das neu hinzugefügte Selektions-Element sichtbar, nachdem das Objekt durch den Befehl WinUpdate() neu gezeichnet wurde. Beim Canvas-Objekt muss zusätzlich die Option WinUpdState angegeben werden. Beim TreeView-Objekt muss das neu hinzugefügte Selektions-Element explizit übertragen werden, damit es im TreeView-Objekt sichtbar wird. Bei Verwendung von WinMsdInsert ist dies standardmäßig der Fall. Durch WinMsdNoTreeSync kann die Übertragung des neuen Elementes in den TreeView verhindert werden. Um alle



Kontakt

Elemente einer Mehrfachselektion in das TreeView-Objekt zu übertragen kann der Befehl WinMsdUpdate() benutzt werden.

Natürlich kann für die Erzeugung eines neuen Selektions-Elements auch der Befehl CteInsert() verwendet werden, jedoch müssen dann die Eigenschaften Name und ID konform mit den Bedingungen der Mehrfachselektion gesetzt werden. Beim TreeView-Objekt wird zudem eine Übertragung der Elemente in der Selektionsmenge in das TreeView-Objekt notwendig. Hierzu kann der Befehl WinMsdUpdate() verwendet werden.

Beispiele:

```
// Einfügen eines Datensatzes$RecList->WinMsdInsert(tblCstCustomer, _WinMsdRecId); // Identisch mi
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Die in (int2) übergebenen Optionen sind nicht zulässig.

ErrFileInvalid Die angegebene Dateinummer in (int1) existiert nicht.

Kontakt

obj ->

WinMsdInsertName(alpha1[, alpha2]) : int



Selektion einer StoList erweitern

obj Deskriptor des Objekts, in dem
selektiert wird

alpha1 Zeichenkette, um die die Selektion
alpha2 erweitert werden soll
Custom-Eigenschaft (optional)

Resultat int	<u>Fehlerwert</u>	Element
	<u>ErrOk</u>	aufgenommen.
	<u>ErrMsdExists</u>	Das Element ist bereits selektiert.

Siehe [Verwandte Befehle,](#)
[WinMsdDeleteName\(\)](#)

WinMsdReadName(), WinMsdInsert()

Mit diesem Befehl wird die Zeichenkette (alpha1) in die Menge der selektierten Elemente hinzugefügt.

Das Objekt, in dem die Selektion stattfindet wird als (obj) der Anweisung übergeben. In (alpha1) wird das Element übergeben, dass in die Selektion aufgenommen werden soll.

In (alpha2) kann eine Zeichenkette angegeben werden, die in der Custom-Eigenschaft des Cte-Items gespeichert wird.

Natürlich kann für die Erzeugung eines neuen Selektions-Elements auch der Befehl CteInsert() verwendet werden, jedoch müssen dann die Eigenschaften Name und ID konform mit den Bedingungen der Mehrfachselektion gesetzt werden.



Dieser Befehl kann nur mit StoList-Objekten verwendet werden.

Beispiele:

```
$StoList->WinMsdInsertName('StoDir');
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Die in (alpha2) übergebene Zeichenkette ist leer.

Kontakt

**obj -> WinMsdRead(bigint1[,
int2]) : handle**



Element einer Selektion lesen
Deskriptor des Objekts, in dem
obj

selektiert wird

bigint1 Element, das gelesen werden soll
oder Dateinummer

Optionen (optional)

int2 WinMsdRecId In (bigint1) wurde
eine Dateinummer
übergeben

Resultat handle Deskriptor des

CteItem-Objektes

Verwandte Befehle,

Siehe WinMsdDelete(), WinMsdInsert(),
WinMsdReadName()

Mit diesem Befehl wird das Element (bigint1) in der Menge der selektierten Elemente gelesen. Mit dem Befehl kann geprüft werden, ob ein bestimmtes Element in der Selektion enthalten ist. Zum Lesen aller selektierten Elemente sollte der Cte-Tree oder die Cte-Liste verwendet werden (siehe SelectionData).

Das Objekt, bei dem die Selektion gelesen werden soll, wird als (obj) der Anweisung übergeben. In (bigint1) wird das Element übergeben, dass gelesen werden soll. In Abhängigkeit vom Objekt muss in (bigint1) eine Datensatz-ID (RecList), eine Zeilennummer (DataList), der Deskriptor eines TreeNode-Objekts (TreeView) oder der Deskriptor eines CanvasGraphic-Objekts (Canvas) übergeben werden.

Die Übergabe einer Datensatz-ID kann auch durch die Angabe der Dateinummer in (bigint1) erfolgen, wenn in (int2) die Konstante WinMsdRecId übergeben wird.

Zurückgegeben wird der Deskriptor auf das CteItem-Objekt, das das selektierte Element enthält (siehe auch SelectionData).

Beispiele:

```
// Überprüfen eines Datensatzes if ($RecList->WinMsdRead($tblCstCustomer, _WinMsdRecId) > 0) { // D
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Die in (int2) übergebenen Optionen sind nicht zulässig.

ErrFileInvalid Die angegebene Dateinummer in (int1) existiert nicht.

Kontakt

obj -> WinMsdReadName(alpha1) : handle  Element
einer Selektion einer StoList lesen
 Deskriptor des Objekts, in dem
obj
 selektiert wird

alpha1 Element,soll das gelesen werden

Resultat handle Deskriptor des Cte-Items

Siehe
Verwandte Befehle,
WinMsdDeleteName(),
WinMsdInsertName(),
WinMsdRead()

Mit diesem Befehl wird das Element (alpha1) in der Menge der selektierten Elemente gelesen. Mit dem Befehl kann geprüft werden, ob ein bestimmtes Element in der Selektion enthalten ist. Zum Lesen aller selektierten Elemente sollte der Cte-Tree oder die Cte-Liste verwendet werden (siehe SelectionData).

Das Objekt, bei dem die Selektion gelesen werden soll, wird als (obj) der Anweisung übergeben. In (alpha1) wird das Element übergeben, dass gelesen werden soll, angegeben.

Zurückgegeben wird der Deskriptor auf das Cte-Item, das das selektierte Element enthält (siehe auch SelectionData).



Dieser Befehl kann nur mit StoList-Objekten verwendet werden.

Beispiele:

```
// Überprüfen, ob das Storage-Objekt StoDir selektiert ist if ($StoList->WinMsdReadName('StoDir'))
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist ungültig.

Kontakt

obj ->
WinMsdUpdate([int1])

: int
Selektion aktualisieren
 Deskriptor des Objekts, in
obj

dem selektiert wird
 Optionen (reserviert /
int1
 optional)

 Fehlerwert

Resultat int ErrOk Element
 aufgenommen.

Siehe Verwandte Befehle

Der Befehl überträgt alle Selektions-Elemente einer Mehrfachselektion in die interne Selektionsliste des TreeView-Objekts. Nach erfolgreicher Durchführung des Befehls sind alle TreeNode-Objekte, die in der Mehrfachselektion vorhanden sind auch im TreeView selektiert. Ein Neuzeichnen des TreeView-Objektes findet nur statt, wenn die Eigenschaft AutoUpdate des TreeView-Objekts gesetzt (true) ist.

Als Objekt muss ein TreeView-Objekt oder SelectionData-Objekt, das zu einem TreeView-Objekt gehört, übergeben werden.

Zur Zeit können keine Optionen angegeben werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der als (obj) übergebene Deskriptor ist kein TreeView-Objekt oder kein SelectionData-Objekt, das zu einem TreeView-Objekt gehört.

ErrValueInvalid Die in (int1) übergebenen Optionen sind nicht zulässig.

Kontakt

Befehle für dynamische Strukturen

Liste der Befehle und Konstanten zur Bearbeitung von dynamischen Strukturen

Befehlsgruppen,
Befehlsliste,

Siehe Dynamische
Strukturen,
Beispiel

Befehle

- CteClear
- CteClose
- CteDelete
- CteInfo
- CteInsert
- CteInsertItem
- CteInsertNode
- CteNodeValueAlpha
- CteOpen
- CteRead

Konstanten

- CteAfter
- CteAttribList
- CteAttribTree
- CteBefore
- CteChildList
- CteChildTree
- CteCmpE
- CteCmpG
- CteCmpGE
- CteCmpL
- CteCmpLE
- CteCount
- CteCustom
- CteFirst
- CteItem
- CteLast
- CteList
- CteNext
- CteNode
- CteNodePath
- CteNodePathCI
- CteNodeValueRead
- CteNodeValueWrite
- CtePrev
- CteSearch
- CteSearchCI
- CteTree
- CteTreeCI

Kontakt

obj ->
CteClear(logic1[,
int2])
Liste/Knoten leeren
obj Liste/Knoten
logic1 Enthaltene Elemente
 löschen
 Knotenart bei Knoten
 (optional)
 CteChild Untergeordnete
int2 Knoten löschen
 CteAttrib Attributknoten
 löschen
 Verwandte Befehle,
Siehe CteDelete(), Beispiel



Mit dieser Funktion können alle Elemente aus einer Liste oder einem Knoten entfernt werden. Wird in (logic1) true angegeben, werden die Elemente gelöscht, bei false bleiben sie nach dem Entfernen erhalten.



Bei einem CteNode-Objekt werden die untergeordneten Knoten bzw. die Attributknoten unabhängig von (logic1) immer gelöscht.

Beispiele:

```
// Liste leeren und Elemente löschenList->CteClear(true); // Liste leeren und Elemente beibehalte
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Liste/Baum (obj) ungültig
ErrValueInvalid Knotenart (int2) unbekannt

Kontakt

obj -> CteClose() |  Element, Liste oder

Knoten löschen

Element,

obj Liste oder

Knoten

Verwandte

Befehle,

Siehe [CteOpen\(\)](#),

[CteClear\(\)](#),

[Beispiel](#)

Mit dieser Funktion wird ein mit [CteOpen\(\)](#) erzeugtes Objekt gelöscht. Beim Löschen einer Liste oder eines Baums bleiben alle darin enthaltenen Elemente erhalten und können wieder in eine andere dynamische Struktur eingefügt werden. Um alle enthaltenen Elemente auf einmal zu löschen, kann [CteClear\(\)](#) benutzt werden.

Beim Löschen eines Elementes wird dieses automatisch aus allen Listen entfernt.

Beim Löschen eines Knotens wird dieser automatisch aus dem übergeordneten Knoten entfernt.

Es ist darauf zu achten, dass jedes nicht mehr verwendete Objekt auch gelöscht wird, da sonst der verfügbare Arbeitsspeicher mit der Zeit immer geringer wird.



Bei einem CteNode-Objekte werden die untergeordneten Knoten bzw. die Attributknoten ebenfalls gelöscht.

Beispiel:

```
// Objekt löschenItem->CteClose();
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Element, Liste oder Knoten (obj) ungültig

Kontakt

obj -> CteDelete(handle1[, int2]) :



logic

Element aus Liste/Knoten entfernen

obj Liste/Knoten

handle1 Element

 Knotenart bei Knoten

 (optional)

CteChild Untergeordneten

int2

 Knoten

 entfernen

CteAttrib Attributknoten

 entfernen

Resultat logic Entfernungserfolg

Verwandte Befehle,

Siehe

CteInsert()

Mit dieser Funktion wird das Element (handle1) aus der Liste oder dem Knoten (obj) entfernt. Das Resultat ist false, wenn sich das Element nicht in der Liste/dem Baum befindet. Das Element selbst bleibt dabei erhalten.

Beispiele:

```
// Element aus Liste entfernenList->CteDelete(tItem); // Untergeordneten Knoten aus Knoten entfer
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Struktur (obj) oder Element (handle1) ungültig

Kontakt

obj -> CteInfo(int1) : int  Listen- / Element- /

Knoteninformation ermitteln

obj Liste / Element / Knoten
Informationstyp
CteCount Elementanzahl von
 Liste ermitteln
int1 CteList Verkettete Liste von
 Element ermitteln
 CteTree Sortierte Liste von
 Element ermitteln

Resultat int Struktur-/Elementinformation

Siehe Verwandte Befehle

Mit dieser Funktion können verschiedene Informationen von einem Knoten, einer verketteten Liste, einer sortierten Liste oder einem Element ermittelt werden.

Bei einer Liste wird bei CteCount die Anzahl der enthaltenen Elemente zurückgeliefert.



Bei CteNode-Objekten wird die Anzahl aller Elemente in der Struktur gezählt.
Sollen nur die direkten Kinder / Attribute gezählt werden, können die Eigenschaften ChildCount und AttribCount verwendet werden.

Für ein Element kann damit ermittelt werden, in welchen Listen es enthalten ist. Das Resultat bei CteList bzw. CteTree ist 0, wenn das Element in keiner Liste enthalten ist.

Beispiel:

```
// Elementanzahl von Liste ermittelntCount # tList->CteInfo(_CteCount); // Verkettete Liste von El
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Liste/Element (obj) ungültig

ErrValueInvalid Informationstyp (int1) unbekannt oder (obj) vom falschen Typ

Kontakt

obj -> CteInsert(handle1[, int2[, handle3]]) : logic



Element in Liste/Baum einfügen

obj Liste/Knoten

handle1 Element

Einfügeposition bei
verketteter Liste und Knoten
(optional)
CteFirst Element am
 Anfang einfügen
CteLast Element am Ende
 einfügen
CteBefore Element vor
 Referenzelement

int2

einfügen
CteAfter Element nach
 Referenzelement
 einfügen

Listenart bei Knoten

(optional)
CteChild Untergeordneten
 Knoten einfügen
CteAttrib Attributknoten

einfügen

handle3 Referenzelement (optional)

Resultat logic Einfügeerfolg
 Verwandte Befehle,

Siehe

CteDelete(), Beispiel

Mit diesem Befehl wird ein Element in eine Liste oder einen Knoten eingefügt. Nach der Anweisung sollte das Resultat überprüft werden. Ist das Resultat false konnte das Element nicht eingefügt werden.

In Listen können alle Objekte außer Storage- und Binäre Objekte eingefügt werden.



Ein Element kann in eine verkettete Liste und/oder eine sortierte Liste eingefügt werden.

- Einfügen in eine verkettete Liste
- Einfügen in eine sortierte Liste
- Einfügen in einen Knoten

Einfügen in eine verkettete Liste

Das Element kann entweder am Anfang (siehe CteFirst) oder am Ende der Liste (siehe CteLast) eingefügt werden. Soll das Element vor oder nach einem bereits in der Liste enthaltenen Element eingefügt werden, wird CteBefore bzw. CteAfter in Verbindung mit dem Referenzelement (handle3) angegeben. Wird (int2) nicht angegeben, erfolgt das Einfügen am Ende der Liste.

Kontakt

Das Resultat ist false, wenn das Element bereits in einer Liste enthalten ist oder sich das Referenzelement nicht in der Liste (obj) befindet (bei Verwendung von CteBefore oder CteAfter).

Beispiele:

```
// Element an Listenanfang einfügenList->CteInsert(tItem, _CteFirst); // Element an Listenende ei
```

Einfügen in eine sortierte Liste

Das Element wird anhand seiner Eigenschaft Name in die Liste eingesortiert. Die Eigenschaft darf nicht leer sein.

Das Resultat ist false, wenn das Element bereits in einer sortierten Liste enthalten ist oder sich bereits ein Element mit gleichem Namen in der Liste befindet.

Beispiel:

```
// Element in Liste einsortierenList->CteInsert(tItem);
```

Einfügen in einen Knoten

Der Knoten wird entweder mit CteChild als untergeordneter Knoten oder mit CteAttrib als Attributknoten eingefügt. Standardmäßig wird CteChild verwendet

Verfügt der Knoten über eine verkettete Liste, können auch die Optionen zur Positionierung verwendet werden.

Das Resultat ist false, wenn der Knoten bereits in einem Knoten enthalten ist oder, wenn der Knoten über eine sortierte Liste verfügt, sich bereits ein Knoten mit gleichem Namen in dem Knoten befindet.

Beispiele:

```
// Untergeordneten Knoten in Knoten einfügenNode->CteInsert(tNode, _CteChild); // Untergeordneten
```

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Liste/Knoten (obj), Element (handle1) oder Referenzelement (handle3) ungültig
<u>ErrValueInvalid</u>	Einfügeposition (int2) unbekannt

Kontakt

obj -> CteInsertItem(alpha1, bigint2,
alpha3[, int4[, int5]] : handle
Element erzeugen und in Liste einfügen
obj Liste
alpha1 Name-Eigenschaft
bigint2 ID-Eigenschaft
alpha3 Custom-Eigenschaft
 Einfügeposition bei verketteter Liste
 (optional)
 CteFirst Element am Anfang einfügen
 CteLast Element am Ende einfügen
int4 CteBefore Element vor Referenzelement
 einfügen
 CteAfter Element nach Referenzelement
 einfügen
int5 Referenzelement (optional)
 ErrExists Element mit dem
 Namen ist bereits
 vorhanden



Resultat handle

Siehe ErrNameInvalid Der Name ist leer
 > 0 Deskriptor des
 Elements
Verwandte Befehle, CteOpen(), CteInsert(),
CteDelete()

Mit diesem Befehl wird ein Element mit den Eigenschaften Name (alpha1), ID (bigint2) und Custom (alpha3) erzeugt und in die Liste (obj) eingefügt.

Das Resultat ist ErrExists, wenn ein Element mit dem Namen bereits in einer sortierten Liste vorhanden ist. Ist der Name leer, ist das Resultat ErrNameInvalid. Andernfalls wird der Deskriptor des eingefügten Elementes zurückgegeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Liste/Knoten (obj), oder Referenzelement (int5) ungültig.

ErrValueInvalid Einfügeposition (int4) unbekannt

Kontakt

obj -> CteInsertNode(alpha1, bigint2, var3[,
int4[, int5[, alpha6]]]) : handle Knoten erzeugen
und in Knoten einfügen



obj Knoten
alpha1 Name-Eigenschaft
bigint2 ID-Eigenschaft
var3 Value-Eigenschaft
Einfügeposition bei verketteter Liste (optional)
CteFirst Element am Anfang einfügen
CteLast Element am Ende einfügen
CteBefore Element vor Referenzelement
int4 einfügen
CteAfter Element nach Referenzelement
einfügen
Listenart (optional)
CteChild Untergeordneten Knoten einfügen
CteAttrib Attributknoten einfügen
int5 Referenzknoten (optional)
alpha6 Custom-Eigenschaft (optional)
ErrExists Knoten mit dem
Namen ist bereits
vorhanden
Resultat handle ErrNameInvalid Der Name ist leer
> 0 Deskriptor des
Knoten
Siehe Verwandte Befehle, CteOpen(), CteInsert(),
CteDelete()

Mit diesem Befehl wird ein Knoten mit den Eigenschaften Name (alpha1), ID (bigint2), Value (var3) und Custom (alpha6) erzeugt und in den Knoten (obj) eingefügt.

Die Eigenschaft Flags des neuen Knotens wird vom Knoten (obj) übernommen.

Der Knoten wird entweder mit CteChild als untergeordneter Knoten oder mit CteAttrib als Attributknoten eingefügt. Standardmäßig wird CteChild verwendet.

Verfügt der Knoten über eine verkettete Liste, können auch die Optionen zur Positionierung verwendet werden.

Das Resultat ist ErrExists, wenn ein Element mit dem Namen bereits in einer sortierten Liste vorhanden ist. Ist der Name leer, ist das Resultat ErrNameInvalid. Andernfalls wird der Deskriptor des eingefügten Knoten zurückgegeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Knoten (obj), oder Referenzknoten (int5) ungültig

ErrValueInvalid Einfügeposition (int4) unbekannt

Kontakt

obj -> CteNodeValueAlpha(handle1, int2) : int 

Zeichenkette des Knoten lesen oder schreiben

obj CteNode-Objekt

handle1 Memory-Objekt

Durchzuführende Operation:

CteNodeValueWrite Inhalt in Knoten

int2

übertragen

CteNodeValueRead Inhalt aus Knoten lesen

Resultat int ErrOK Befehl erfolgreich durchgeführt 

Siehe Verwandte Befehle, CteOpen(), CteInsert(),

CteDelete()

Die Länge eines CteNode-Objektes mit alphanumerischem Inhalt (Type = TypeAlpha) ist nicht längenlimitiert. Mit der Eigenschaft ValueAlpha lassen sich jedoch maximal 65520 Byte schreiben bzw. lesen. CteNodeValueAlpha() schreibt und liest alphanumerische Inhalte mit mehr als 65520 Byte.

Der Deskriptor des CteNode-Objektes wird im Argument (obj) angegeben. Das Argument (handle1) enthält den Deskriptor des Memory-Objektes. Das Argument (int2) definiert die durchzuführende Operation:

CteNodeValueWrite Diese Option kopiert den Inhalt des Memory-Objektes in das CteNode-Objekt. Nach dem Aufruf kann auch die Eigenschaft ValueAlpha verwendet werden, um den neuen Inhalt auszulesen. Hier können jedoch maximal 65520 Byte verarbeitet werden. Der Datentyp (Eigenschaft Type) des CteNode-Objektes wird auf TypeAlpha modifiziert.

CteNodeValueRead Diese Option kopiert den Inhalt des CteNode-Objektes in das Memory-Objekt. Der Datentyp (Eigenschaft Type) des CteNode-Objektes muss TypeAlpha sein.

 Der Zeichensatz des Memory-Objektes (Eigenschaft Charset) wird ignoriert. Der Rückgabewert liefert immer ErrOk. Konnte die Verarbeitung nicht durchgeführt werden, wird ein Laufzeitfehler generiert.

Mögliche Laufzeitfehler:

ErrHdlInvalid CteNode-Objekt (obj) oder Memory-Objekt (handle1) enthalten keine gültigen Deskriptoren.

ErrValueInvalid Der Modus (int2) enthält einen ungültigen Wert.

Der Buffer des Memory-Objektes ist nicht groß genug, um den Inhalt komplett speichern zu können. Bei Verwendung der Option MemAutoSize kann der Fehler auftreten, wenn das

ErrValueRange Memory-Objekt mehr als 512 MB an Daten benötigt (nur 32-bit-Client). Die Fehler treten beim Lesen des Inhaltes mit CteNodeValueRead auf.

ErrFldType Der Datentyp des CteNode-Objektes ist nicht TypeAlpha. Der Fehler tritt beim Lesen des Inhaltes mit CteNodeValueRead auf.

ErrMemExhausted

Kontakt

Nicht genügend freier Arbeitsspeicher zum Durchführen des Befehls.

Kontakt



CteOpen(int1[, int2]) : handle Element,

Liste oder Knoten erzeugen

Objekttyp

CteItem Element erzeugen

CteList Verkettete Liste erzeugen

CteTree Sortierte Liste erzeugen

int1

CteTreeCI Sortierte Liste ohne

Unterscheidung von
Groß-/Kleinschreibung

erzeugen

CteNode Knoten erzeugen

Optionen bei Knoten (optional)

CteChildList Verkettete Liste der

untergeordneten Knoten

CteChildTree Sortierte Liste der

untergeordneten Knoten

CteChildTreeCI Sortierte Liste der

untergeordneten Knoten

(ohne Unterscheidung

der

int2

CteAttribList Groß-/Kleinschreibung)

Verkettete Liste der

Attributknoten

CteAttribTree Sortierte Liste der

Attributknoten

CteAttribTreeCI Sortierte Liste der

Attributknoten (ohne

Unterscheidung der

Groß-/Kleinschreibung)

Resultat handle Element, Liste oder Knoten

Verwandte Befehle, CteClose(),

Siehe

CteInsertItem(), CteInsertNode(), Beispiel

Mit dieser Funktion wird ein neues Objekt einer dynamischen Struktur erzeugt. Der Typ des erzeugten Objekts wird in (int1) angegeben.

Die Eigenschaften eines neuen Objektes sind leer bzw. NULL.

Beispiele:

```
// Element erzeugenItem # CteOpen(_CteItem); // Verkettete Liste erzeugenList # CteOpen(_CteList
```

Bei Knoten können Optionen (int2) übergeben werden. Werden keine Optionen angegeben, werden die Optionen CteChildList | CteAttribList | CteAttribTree verwendet. Um einen Knoten ohne Listen zu erzeugen muss 0 übergeben werden.

Die Optionen werden in die Eigenschaft Flags übernommen.

Beispiele:

Kontakt

// Knoten mit verkettete Liste für untergeordnete Knoten// und verketteter und sortierter Liste

Die Funktion gibt den Deskriptor des erzeugten Objekts zurück. Konnte das Objekt nicht angelegt werden, gibt der Befehl 0 zurück.

Mögliche Laufzeitfehler:

ErrValueInvalid Objekttyp (int1) oder Optionen (int2) unbekannt

Kontakt

```
obj -> CteRead(int1[,
handle2[, alpha3]]):
handle
Element/Knoten lesen
obj      Liste/Knoten
int1    Suchmodus (siehe Text)
handle2 Referenzelement (optional)
alpha3   Suchname (optional)
Resultat handle Element-Deskriptor
          Verwandte Befehle,
Siehe    CteInsert(), CteDelete(),
          Beispiel
```



Mit diesem Befehl können Elemente in einer dynamischen Struktur gelesen werden.

Folgende Suchmodi (int1) sind sowohl für Listen (CteList und CteTree) als auch für Knoten (CteNode) möglich:

- CteFirst

Erstes Element lesen

- CteLast

Letztes Element lesen

- CtePrev

Element vor Referenzelement lesen

- CteNext

Element nach Referenzelement lesen

Soll eine Liste eines CteNode-Objekts durchsucht werden, muss der Suchmodus mit einer der folgenden Konstanten kombiniert werden:

- CteChildList

verkettete Liste der untergeordneten Knoten durchsuchen

- CteChildTree

Sortierte Liste der untergeordneten Knoten durchsuchen

- CteAttribList

verkettete Liste der Attribute durchsuchen

- CteAttribTree

Sortierte Liste der Attribute durchsuchen

Ist bei CtePrev oder CteNext das Referenzelement nicht angegeben oder nicht in der Liste/dem Knoten (obj) enthalten, ist das Resultat 0.

Beispiel:

Kontakt

```
// Alle Elemente einer Liste durchlaufen for tItem # tList->CteRead(_CteFirst); loop tItem # tLi
```

Diese Operationen können mit CteSearch oder CteSearchCI erweitert werden. Dabei wird der Name des Elements zusätzlich auf Ähnlichkeit mit (alpha3) überprüft und dadurch nach dem ersten Element gesucht, das dem Kriterium entspricht. Durch die zusätzliche Angabe von CteCustom kann anstatt der Eigenschaft Name die Eigenschaft Custom überprüft werden.

Beispiele:

```
// Erstes Element mit Namen 'Test' lesen tItem # tList->CteRead(_CteFirst | _CteSearch, 0, 'test')
```

Der direkte Zugriff über den Namen ist nur bei sortierten Listen (CteTree) und Knoten (CteNode) möglich. Dabei wird entweder der Name im Referenzelement (int2) oder der Name in (alpha3) als Suchwert verwendet, wobei (alpha3) Vorrang hat.

Folgende Suchmodi (int1) sind für sortierte Listen und Knoten mit sortierten Listen möglich:

- CteCmpL

Element mit nächstkleinerem Namen lesen

- CteCmpLE

Element mit gleichen oder nächstkleinerem Namen lesen

- CteCmpE

Element mit gleichen Namen lesen

- CteCmpGE

Element mit gleichen oder nächstgrößerem Namen lesen

- CteCmpG

Element mit nächstgrößerem Namen lesen



Diese Modi können nicht mit CteSearch, CteSearchCI und CteCustom kombiniert werden.

Beispiele:

```
// Element mit Namen von tRefItem lesen tItem # tList->CteRead(_CteCmpE, tRefItem); // Erstes Eleme
```

Bei Knoten muss neben dem Suchmodus auch die zu durchsuchende Liste angebenen werden:

- CteChildList

Suche in verketteter Liste der untergeordneten Knoten

- CteChildTree

Suche in sortierter Liste der untergeordneten Knoten

- CteAttribList

Kontakt

Suche in verketteter Liste der Attributknoten

- CteAttribTree

Suche in sortierter Liste der Attributknoten

Beispiele:

```
// Ersten Knoten in verketteter Liste der untergeordneten Knoten suchen.tNode # tNodeRoot->CteRea
```

Für Bäume besteht zusätzlich die Möglichkeit mit CteNodePath oder

CteNodePathCI einen Knoten über mehrere Ebenen zu suchen. Dabei wird immer die sortierte Liste des Knotens verwendet. Falls keine sortierte Liste vorhanden ist, wird die verkettete Liste durchsucht. Die Suchoptionen werden in der letzten Ebene angewendet.

Die Separatoren für die Suche mit CteNodePath und CteNodePathCI können über die Eigenschaften CteNodeSepPath und CteNodeSepAttrib des Sys-Objekts eingestellt werden.

 Diese Modi können nicht mit CteCmpL, CteCmpLE, CteCmpE, CteCmpGE und CteCmpG kombiniert werden.

Beispiele:

```
// Untergeordneten Knoten mit Namen "child" im untergeordneten Knoten mit Namen "parent" suchen.t
```

Das Resultat bei allen Leseoperationen ist 0, wenn kein Element gefunden wurde, sonst wird der Deskriptor des gefundenen Elements zurückgegeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Liste/Knoten (obj) oder Element (int2) ungültig

ErrValueInvalid Suchmodus (int1) unbekannt

Konstanten für dynamische Strukturen

Konstanten für dynamische Strukturen

Befehle für

Siehe dynamische
Strukturen

- CteAfter
- CteAttribList
- CteAttribTree
- CteBefore
- CteChildList
- CteChildTree
- CteCmpE
- CteCmpG
- CteCmpGE
- CteCmpL
- CteCmpLE
- CteCount
- CteCustom
- CteFirst
- CteItem
- CteLast
- CteList
- CteNext
- CteNode
- CteNodePath
- CteNodePathCI
- CtenodeValueRead
- CtenodeValueWrite
- CtePrev
- CteSearch
- CteSearchCI
- CteTree
- CteTreeCI

Kontakt

_CteAfter

Element nach Referenzelement einfügen

Wert 6 / 0x0006

Verwandte

Siehe Befehle,

CteInsert(),

CteBefore

Option bei CteInsert(), CteInsertItem() und CteInsertNode() durch die ein Element nach dem Referenzelement in eine Liste/Knoten eingefügt werden kann.

Kontakt

_CteAttrib

Attributknoten

Wert 2.097.152 /
0x00200000

Verwandte

Befehle,

Siehe CteInsertNode(),
CteDelete(),
CteClear()

Wird bei einer Anweisung diese Option angegeben, betrifft die Anweisung die Attributknoten.

_CteAttribList

Verkettete Liste der Attributknoten

Wert 65.536 /
0x00010000

Verwandte

Siehe Befehle,

CteOpen(),
CteRead()

Wird diese Option bei der Anweisung CteOpen() angegeben, wird ein Knoten-Objekt mit einer verketteten Liste für Attributknoten erzeugt.

Bei der Übergabe bei der Anweisung CteRead(), wird die verkettete Liste der Attributknoten durchsucht.

Kontakt

_CteAttribTree

Sortierte Liste der Attributknoten

Wert 131.072 /
0x00020000

Verwandte

Siehe Befehle,

CteOpen(),
CteRead()

Wird diese Option bei der Anweisung CteOpen() angegeben, wird ein Knoten-Objekt mit einer sortierten Liste für Attributknoten erzeugt.

Bei der Übergabe bei der Anweisung CteRead(), wird die sortierte Liste der Attributknoten durchsucht.

Kontakt

_CteAttribTreeCI

Sortierte Liste der Attributknoten ohne Unterscheidung der Groß-/Kleinschreibung

Wert 393.216 /

0x00060000

Verwandte

Siehe Befehle,

CteOpen()

Option bei CteOpen(). Der Knoten wird mit einer sortierten Liste für Attributknoten erzeugt. In der Liste findet keine Unterscheidung der Groß-/Kleinschreibung statt.

Kontakt

_CteBefore

Element vor Referenzelement einfügen

Wert 5 / 0x0005

Verwandte

Siehe Befehle,

CteInsert(),

CteAfter

Option bei CteInsert(), CteInsertItem() und CteInsertNode() durch die ein Element vor dem Referenzelement in eine Liste/Knoten eingefügt werden kann.

Kontakt

_CteChild

Untergeordnete Knoten

Wert 1.048.576 /
0x00100000

Verwandte

Befehle,

Siehe [CteInsertNode\(\)](#),
[CteDelete\(\)](#),
[CteClear\(\)](#)

Wird bei einer Anweisung diese Option angegeben, betrifft die Anweisung die untergeordneten Knoten.

Kontakt

_CteChildList

Verkettete Liste der untergeordneten Knoten

Wert 4.096 /
0x00001000

Verwandte

Siehe Befehle,

CteOpen(),
CteRead()

Wird diese Option bei der Anweisung CteOpen() angegeben, wird ein Knoten-Objekt mit einer verketteten Liste für untergeordnete Knoten erzeugt.

Bei der Übergabe bei der Anweisung CteRead(), wird die verkettete Liste der untergeordneten Knoten durchsucht.

Kontakt

_CteChildTree

Sortierte Liste der untergeordneten Knoten

Wert 8.192 /

0x00002000

Verwandte

Siehe Befehle,

CteOpen(),

CteRead()

Wird diese Option bei der Anweisung CteOpen() angegeben, wird ein Knoten-Objekt mit einer sortierten Liste für untergeordnete Knoten erzeugt.

Bei der Übergabe bei der Anweisung CteRead(), wird die sortierte Liste der untergeordneten Knoten durchsucht.

Kontakt

_CteChildTreeCI

Sortierte Liste der untergeordneten Knoten ohne Unterscheidung der Groß-

/Kleinschreibung

^{24.576 /}

Wert

Verwandte

Siehe Befehle,

CteOpen()

Option bei CteOpen(). Der Knoten wird mit einer sortierten Liste für untergeordnete Knoten erzeugt. In der Liste findet keine Unterscheidung der Groß-/Kleinschreibung statt.

Kontakt

_CteCmpE

Element mit gleichem Namen lesen

Wert 13 /

0x000D

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die das Element mit dem gleichen Namen gelesen werden kann.

Kontakt

_CteCmpG

Element mit nächstgrößerem Namen lesen
Wert 15 /

0x000F

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die das Element mit dem nächstgrößeren Namen gelesen werden kann.

Kontakt

_CteCmpGE

Element mit gleichem oder nächstgrößerem Namen lesen
Wert 14 /

0x000E

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die das Element mit dem gleichen oder nächstgrößeren Namen gelesen werden kann.

Kontakt

_CteCmpL

Element mit nächstkleinerem Namen lesen

Wert 11 /

0x000B

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die das Element mit dem nächstkleineren Namen gelesen werden kann.

Kontakt

_CteCmpLE

Element mit gleichem oder nächstkleinerem Namen lesen
Wert 12 /

0x000C

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die das Element mit dem gleichen oder nächstkleineren Namen gelesen werden kann.

Kontakt

_CteCount

Elementanzahl einer Liste oder eines Knotens ermitteln
Wert $16 /$

0x0010

Verwandte

Siehe Befehle,

CteInfo()

Option bei CteInfo() durch die die Anzahl der Elemente einer Liste ermittelt werden kann.



Bei CteNode-Objekten wird die Anzahl aller Elemente in der Struktur gezählt.

Sollen nur die direkten Kinder / Attribute gezählt werden, können die Eigenschaften ChildCount und AttribCount verwendet werden.

Kontakt

_CteCustom

Element über Custom-Eigenschaft suchen

Wert 1.024 /

0x00000400

Verwandte

Siehe Befehle,

CteRead(),

CteSearch

Option bei CteRead() durch die ein Element über die Eigenschaft Custom gesucht werden kann.

Kontakt

_CteFirst

Erstes Element

Wert 1 / 0x0001

Verwandte

Befehle,

Siehe CteInsert(),

CteRead(),

CteBefore

Option bei CteInsert() und CteRead() durch die ein Element am Anfang der Liste eingefügt bzw. das erste Element einer Struktur gelesen werden kann.

Kontakt

_CteItem

Element erzeugen

Wert 0 / 0x0000

Verwandte

Siehe Befehle,

CteOpen()

Option bei CteOpen() durch die ein neues Element erzeugt wird. Das Objekt besitzt folgende Eigenschaften:

- Name
- ID
- Custom

Das Objekt kann ebenfalls mit den Anweisungen CteInsertItem() erzeugt werden.

Kontakt

_CteLast

Letztes Element

Wert 2 / 0x0002

Verwandte

Befehle,

Siehe CteInsert(),

CteRead(),

CteAfter

Option bei CteInsert() und CteRead() durch die ein Element am Ende der Liste eingefügt bzw. das letzte Element einer Struktur gelesen werden kann.

Kontakt

_CteList

Verkettete Liste erzeugen/ermitteln

Wert 1 / 0x0001

Verwandte

Siehe Befehle,

CteOpen(),

CteInfo()

Option bei CteOpen() durch die eine neue verkettete Liste erzeugt wird. Das Objekt besitzt folgende Eigenschaften:

- Name
- ID
- Custom

Kontakt

_CteNext

Element nach Referenzelement lesen

Wert 4 / 0x0004

Verwandte

Siehe Befehle,

CteRead(),

CtePrev

Option bei CteRead() durch die das Element nach dem Referenzelement gelesen werden kann.

Kontakt

_CteNode

Knoten erzeugen

Wert 4 / 0x0004

Verwandte

Siehe Befehle,

CteOpen()

Option bei CteOpen() durch die ein neuer Knoten erzeugt wird. Der Knoten besitzt folgende Eigenschaften:

- Name
- ID
- Custom
- Flags
- Parent
- Type
- Value...
- AttribCount
- ChildCount

Nähere Informationen befinden sich im Abschnitt Dynamische Strukturen.

Kontakt

_CteNodePath

Knoten über Pfad aus Name-Eigenschaft suchen Wert 7 /
0x0007

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die ein Knoten über den Pfad aus der Eigenschaft Name gesucht werden kann.

Die Separatoren für Pfad und Attribut können über die Eigenschaften CteNodeSepPath und CteNodeSepAttrib des Sys-Objekts gesetzt werden.

Kontakt

_CteNodePathCI

Knoten über Pfad aus Name-Eigenschaft ohne Unterscheidung der Groß-/Kleinschreibung suchen

Wert 8 / 0x0008

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die ein Knoten über den Pfad aus der Eigenschaft Name ohne Unterscheidung der Groß-/Kleinschreibung gesucht werden kann.

Die Separatoren für Pfad und Attribut können über die Eigenschaften CteNodeSepPath und CteNodeSepAttrib des Sys-Objekts gesetzt werden.

Kontakt

_CteNodeValueRead

Daten in Memory-Objekte einlesen

Wert 0 / 0x0000

[Verwandte Befehle](#),

Siehe [CteNodeValueAlpha\(\)](#),

[CteNodeValueWrite](#)

Option bei [CteNodeValueAlpha\(\)](#) durch die Daten eines [CteNode](#)-Objektes in ein [Memory](#)-Objekt kopiert werden können.

Kontakt

_CteNodeValueWrite

Daten des Memory-Objektes schreiben

Wert 1 / 0x0001

[Verwandte Befehle](#),

Siehe [CteNodeValueAlpha\(\)](#),

[CteNodeValueRead](#)

Option bei [CteNodeValueAlpha\(\)](#) durch die Daten eines Memory-Objektes in ein CteNode-Objekt kopiert werden können.

Kontakt

_CtePrev

Element vor Referenzelement lesen

Wert 3 / 0x0003

Verwandte

Siehe Befehle,

CteRead(),

CteNext

Option bei CteRead() durch die das Element vor dem Referenzelement gelesen werden kann.

Kontakt

_CteSearch

Element über Name- oder Custom-Eigenschaft suchen
Wert 256 /

0x00000100

Verwandte

Siehe Befehle,

CteRead()

Option bei CteRead() durch die ein Element über die Eigenschaft Name gesucht werden kann.

Wird zusätzlich die Option CteCustom angegeben, findet die Suche über die Eigenschaft Custom statt.

Kontakt

_CteSearchCI

Element über Name- oder Custom-Eigenschaft suchen (ohne Unterscheidung der Groß-/Kleinschreibung)
^{512 /}
Wert

Verwandte

Befehle,

Siehe CteRead(),

CteSearch,

CteCustom,

Option bei CteRead() durch die ein Element über die Eigenschaft Name, ohne Unterscheidung der Groß-/Kleinschreibung, gesucht werden kann.

Kontakt

_CteTree

Sortierte Liste erzeugen/ermitteln

Wert 2 / 0x0002

Verwandte

Siehe Befehle,

CteOpen(),

CteInfo()

Option bei CteOpen() durch die eine neue sortierte Liste erzeugt wird. Das Objekt besitzt folgende Eigenschaften:

- Name
- ID
- Custom

Kontakt

_CteTreeCI

Baum ohne Unterscheidung von Groß-/Kleinschreibung erzeugen Wert 3 / 0x0003

Verwandte

Siehe Befehle,

CteOpen()

Option bei CteOpen() durch die eine neue sortierte Liste erzeugt wird. Das Objekt besitzt folgende Eigenschaften:

- Name
- ID
- Custom

In dieser sortierten Liste werden Groß- und Kleinbuchstaben nicht unterschieden.

Kontakt

Befehle für Memory-Objekte

Liste der Befehle und Konstanten zur Bearbeitung von Memory-Objekten
Befehlsgruppen,

Siehe Befehlsliste,

Memory

Befehle

- BinReadMem
- BinWriteMem
- FsiReadMem
- FsiWriteMem
- MemAllocate
- MemCnv
- MemCompress
- MemCopy
- MemDecrypt
- MemEncrypt
- MemFindByte
- MemFindStr
- MemFree
- MemGenKeyPair
- MemHash
- MemHMAC
- MemReadByte
- MemReadStr
- MemResize
- MemSign
- MemUncompress
- MemVerify
- MemWriteByte
- MemWriteStr
- MsxReadMem
- MsxWriteMem
- PdfTextExtractMem
- SckReadMem
- SckWriteMem
- WinRtfPicInsertMem

Konstanten

- ComprFmtDeflate
- ComprFmtGzip
- ComprFmtZlib
- ComprLvlDefault
- MemAppend
- MemCipherAES128
- MemCipherAES192
- MemCipherAES256
- MemCipherModeCBC
- MemCipherModeCTR
- MemCipherModeGCM

- [MemCipherModeOFB](#)
- [MemCipherNoPadding](#)
- [MemCipherRSA](#)
- [MemDataLen](#)
- [MemDecBase64](#)
- [MemDecHex](#)
- [MemEncBase64](#)
- [MemEncHex](#)
- [MemIVBase64](#)
- [MemIVHex](#)
- [MemIVMem](#)
- [MemKeyAsymPrivate](#)
- [MemKeyAsymPublic](#)
- [MemKeyBase64](#)
- [MemKeyHex](#)
- [MemKeyMem](#)
- [MemObjSize](#)
- [MemPaddingRSAOaep](#)
- [MemPaddingRSAPkcs1](#)
- [MemSignatureBase64](#)
- [MemSignatureHex](#)
- [MemSignDSA](#)
- [MemSignRSA](#)

Kontakt

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)

BinErrorDecryption Eindeutiger

int3 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 Objektinhalt verschlüsselt gespeichert wurde, muss in (alpha2) der entsprechende Verschlüsselungscode angegeben werden. Bei einem inkorrektten 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.

Kontakt

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



Binäres Objekt aus Memory-Objekt schreiben
Deskriptor eines

obj binären Objekts

handle1 Deskriptor eines
Memory-Objekts
Kompressionsstufe

int2 (optional)
Verschlüsselungs-Code

alpha3 (optional) —

Siehe Verwandte Befehle,
BinReadMem()

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

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

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

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

ErrBinNoLock Das binäre Objekt ist nicht exklusiv gesperrt.

ErrBinNoData Das Memory-Objekt enthält keine Daten rDeadlock

Verklemmung aufgetreten

Mögliche Laufzeitfehler:

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

Kontakt

obj -> FsiReadMem(handle1, int2,
int3) : int



Datei in Memory-Objekt lesen

obj Datei-Deskriptor

handle1 Deskriptor des Memory-Objekts

int2 Position im Memory-Objekt

int3 Anzahl der Bytes
Anzahl der gelesenen Bytes

Resultat int

oder Fehlerwert



Siehe [Verwandte Befehle](#), [FsiWriteMem\(\)](#)

Mit dieser Funktion werden Daten aus der externen Datei (obj) ab der aktuellen Position gelesen (siehe [FsiSeek\(\)](#) bzw. [FsiSeek64\(\)](#)). In (handle1) muss der Deskriptor eines Memory-Objekts angegeben werden. Aus der Datei werden maximal (int3) Bytes gelesen und ab der Position (int2) in das Memory-Objekt übertragen. Gegebenenfalls wird der Wert der Eigenschaft Len erhöht.

Werden aus der externen Datei Zeichenketten in das Objekt gelesen, muss die Eigenschaft Charset des Memory-Objekts auf den Zeichensatz der externen Datei gesetzt werden, damit die Zeichenketten korrekt verarbeitet werden können. Eine Konvertierung der Zeichenkodierung aufgrund der Angaben bei [FsiOpen\(\)](#) findet nicht statt.

Das Resultat gibt die Anzahl der gelesenen Bytes zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert ([ErrFsi...](#)). Der Fehlerwert des Betriebssystems kann über die Eigenschaft [FsiError](#) abgefragt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Datei-Descriptor (obj) oder der Deskriptor des Memory-Objekts ist ungültig.

ErrValueRange Die übergebenen Werte in (int2) oder (int3) sind ungültig.

Kontakt

obj -> FsiWriteMem(handle1, int2,
int3) : int
Memory-Objekt in Datei schreiben
obj Datei-Deskriptor
handle1 Deskriptor des Memory-Objekts
int2 Position im Memory-Objekt
int3 Anzahl der zu schreibenden Bytes
Anzahl der geschriebenen Bytes
Resultat int oder Fehlerwert



Siehe [Verwandte Befehle, FsiOpen\(\)](#)

Mit dieser Funktion werden Daten in die externe Datei (obj) ab der aktuellen Position geschrieben (siehe [FsiSeek\(\)](#) bzw. [FsiSeek64\(\)](#)). In (handle1) muss der Deskriptor eines Memory-Objekts angegeben werden. Aus dem Memory-Objekt werden ab der Position (int2) eine Anzahl von (int3) Bytes in die Datei geschrieben.

Sollen Zeichenketten in die externen Datei geschrieben, muss die Eigenschaft Charset des Memory-Objekts auf den Zeichensatz der externen Datei gesetzt werden, damit die Zeichenketten beim Einfügen in das Memory-Objekt (siehe [MemWriteStr\(\)](#)) korrekt verarbeitet werden können. Eine Konvertierung der Zeichenkodierung aufgrund der Angaben bei [FsiOpen\(\)](#) findet nicht statt.

Das Resultat gibt die Anzahl der geschrieben Bytes zurück. Ist das Resultat negativ ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert ([ErrFsi...](#)). Der Fehlerwert des Betriebssystems kann über die Eigenschaft [FsiError](#) abgefragt werden.

Mögliche Laufzeitfehler:

- [ErrHdlInvalid](#) Datei-Descriptor (obj) oder der Deskriptor des Memory-Objekts ist ungültig.
- [ErrValueRange](#) Der in (int2) oder (int3) übergebene Wert ist außerhalb des zulässigen Bereichs.

Kontakt

MemAllocate(int1) :



handle

Memory-Objekt erzeugen

Speichermenge in Byte oder
int1

_MemAutoSize

Deskriptor auf das

Resultat handle Objekt

Verwandte Befehle, Memory,

Siehe

[MemFree\(\)](#)

Mit dieser Funktion wird ein **Memory-Objekt** angelegt. Die Größe muss dabei im Bereich von 1 Byte bis 512 MB (bei 32-Bit-Prozessen) bzw. 2 GB (bei 64-Bit-Prozessen) liegen. Der Speicherbereich wird im Adressraum des aktuellen Prozesses angelegt. Der Adressraum ist bei 32-Bit-Prozessen normalerweise 2 GB groß, in speziellen Fällen auch mehr (siehe [ProcessMemoryLimitMB](#)). Als Rückgabewert wird ein Deskriptor auf das Objekt zurückgegeben. Konnte das Objekt nicht angelegt werden, weil nicht ausreichend Speicher vorhanden ist, wird [ErrOutOfMemory](#) zurückgegeben.

Bei der Größe des Objekts können anstelle der Anzahl der Bytes auch folgende Konstanten angegeben werden:

_Mem1K	_Mem1M
_Mem2K	_Mem2M
_Mem4K	_Mem4M
_Mem8K	_Mem8M
_Mem16K	_Mem16M
_Mem32K	_Mem32M
_Mem64K	_Mem64M
_Mem128K	_Mem128M
_Mem256K	_Mem256M
_Mem512K	_Mem512M

Anstelle der Größe des Objekts kann die Konstante **_MemAutoSize** angegeben werden. Die Größe des Objekts wird dann automatisch vergrößert, wenn entsprechende Daten in das Objekt geschrieben werden. Beim Vergrößern des Objekts muss der Speicherinhalt kopiert werden, daher dauert das Schreiben dann länger als bei Objekten mit statischer Größe. Eine Verkleinerung des Objekts findet nicht automatisch statt.

Mit dem Befehl **MemResize()**, kann die Größe nachträglich verändert werden.

Um das Objekt wieder zu entfernen und den Speicher freizugeben, muss die Anweisung **MemFree()** verwendet werden.

Folgende Fehlerwerte sind möglich:

ErrValueInvalid Als Größe (int1) wurde ein Wert ≤ 0 angegeben.

ErrLimitExceeded Bei einem 32-Bit-Prozess wurde eine Größe (int1) von mehr als 512 MB angegeben.

Kontakt

ErrOutOfMemory Der Speicher konnte nicht angefordert werden.

Kontakt

obj -> MemCnv(handle1,



int2) : int

Memory-Objekt konvertieren

obj Deskriptor des Quell-Memory-Objekts

handle1 Deskriptor des Ziel-Memory-Objekts

int2 Art der Konvertierung (siehe Text)

Fehlerwert

int ErrOk Kein Fehler

Resultat int ErrData Ungültige Zeichen im
Quell-Memory-Objekt



Siehe Verwandte Befehle

Diese Funktion überträgt und konvertiert den Inhalt des Memory-Objekts (obj) in den Speicherbereich des Memory-Objekts (handle1).



Das Ziel-Memory-Objekt (handle1) muss mindestens 1024 Byte groß sein.

Alternativ kann es auch mit MemAutoSize angelegt werden.

Mit (int2) kann die Art der Konvertierung angegeben werden:

- Zeichensatzumwandlung

Wird ein Zielzeichensatz angegeben (siehe Charset), erfolgt eine Zeichensatzumwandlung. Als Quellzeichensatz wird die Eigenschaft Charset des Quell-Objekts (obj) verwendet. Es kann zwischen allen unterstützten Zeichensätzen konvertiert werden. Dabei werden Zeichen, die nicht im Zielzeichensatz darstellbar sind, durch ein Fragezeichen ersetzt. Die Eigenschaft Charset des Ziel-Objekts (handle1) wird auf den übergebenen Wert gesetzt. Das Resultat ist immer ErrOk. Zeichen, die nicht umgewandelt werden können werden als Fragezeichen konvertiert.

- Kodierung nach Base64

Durch Angabe der Konstanten MemEncBase64 wird der Inhalt in Base64 kodiert. Die Datenmenge wächst dabei um ein Drittel (33,33%). Es können beliebige binäre Daten kodiert werden.

- Dekodierung aus Base64

Mit der Konstanten MemDecBase64 wird der Inhalt aus Base64 dekodiert. Der Inhalt muss aus gültigen Base64-Daten bestehen. Beim Dekodieren werden Whitespace-Zeichen, wie Leerzeichen und Zeilenumbrüche, ignoriert. Bei nicht erlaubten Zeichen wird der Fehlerwert ErrData zurückgegeben. Die Datenmenge schrumpft dabei um ein Viertel (25%).

- Kodierung in hexadezimale Zeichen

Durch Angabe der Konstanten MemEncHex wird der Inhalt in hexadizimale Zeichen kodiert. Die Datenmenge wächst dabei auf das doppelte. Es können beliebige binäre Daten kodiert werden.

- Dekodierung aus hexadezimalen Zeichen

Mit der Konstanten MemDecHex wird der Inhalt aus hexadezimalen dekodiert. Der Inhalt muss aus gültigen Daten bestehen. Beim Dekodieren werden Whitespace-Zeichen, wie Leerzeichen und Zeilenumbrüche, ignoriert. Bei nicht

Kontakt

erlaubten Zeichen wird der Fehlerwert ErrData zurückgegeben. Die Datenmenge schrumpft dabei um die Hälfte (50%).

Die Eigenschaft Len des Zielobjekts wird bei allen Operationen neu gesetzt.

Resultat

Als Resultat wird ErrOk zurückgegeben, wenn die Konvertierung erfolgreich war. Bei Konvertierungen mit nicht erlaubten Zeichen wird der Fehlerwert ErrData zurückgegeben.

Beispiel:

```
tMemOrg # MemAllocate(_MemAutoSize); tMemOrg->MemWriteStr(1, "..."); tMemcpy # MemAllocate((tMem
```

Mögliche Laufzeitfehler:

ErrHdlInvalid

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

ErrValueInvalid

In (int2) wurde kein zulässiger Wert übergeben.

ErrStringOverflow

Das Zielobjekt ist bei der Zeichensatzumwandlung nicht groß genug (Size), um das umgewandelte Objekt aufzunehmen. Es muss mindestens 1024 Byte groß sein oder über automatische Vergrößerung (MemAutoSize) verfügen.

Kontakt

obj -> MemCompress(int1[,
int2[, int3[, int4[, handle5[,
int6]]]]]) : int



Speicherbereich komprimieren
obj Quelle / Ziel (Memory-Objekt)

Kompressionsformat
ComprFmtDeflate DEFLATE-Format

int1 ComprFmtGzip GZIP-Format
 ComprFmtZlib ZLIB-Format

int2 Kompressionsstufe (optional)

int3 Quellposition (optional)

int4 Quelllänge (optional)

handle5 Ziel (Memory-Objekt, optional)

int6 Zielposition (optional)

Resultat int Fehlerwert



Siehe Verwandte Befehle,

MemUncompress(),

FsiFileCompress()

Dieser Befehl komprimiert den Inhalt des Memory-Objektes (obj).

Es muss eines der folgenden Kompressionsformate (int1) angegeben werden:

ComprFmtDeflate DEFLATE-Format

ComprFmtGzip GZIP-Format

ComprFmtZlib ZLIB-Format

Als Kompressionsstufe (int2) können Werte zwischen 0 (keine Komprimierung) und 9 (maximale Komprimierung) angegeben werden. Alternativ wird mit ComprLvlDefault die Standard-Komprimierungsstufe angegeben.

Im Parameter (int3) kann die Quellposition angegeben werden. Ist dieser Wert nicht angegeben oder 0, werden die Daten ab Beginn des Memory-Objektes komprimiert.

Der Parameter (int4) gibt die zu komprimierende Länge an. Ist dieser Wert nicht angegeben, 0 oder MemDataLen wird der restliche Inhalt (nach der Quellposition) des Memory-Objektes komprimiert.

Optional kann im Parameter (handle5) ein Ziel-Memory-Objekt angegeben werden. Ist dieses nicht angegeben oder ist es identisch mit dem Quellobjekt (obj), wird in das Quellobjekt geschrieben.

Hierbei werden alle Daten verworfen, die hinter dem Dateifuß des Kompressionsformats vorhanden waren.

Zusätzlich kann eine Zielposition (int6) angegeben werden, wenn nicht an den Anfang des Ziel-Objektes geschrieben werden soll. Alle vorhandenen Daten ab der Position werden überschrieben.

Beispiele

```
// Inhalt des Memory-Objektes tMemSrc in neues Memory-Objekt im GZIP-Format komprimiertMemSrc->
```

Kontakt

Fehlerwerte

Folgende Fehlerwerte können von der Funktion zurückgegeben werden:

ErrOk Kein Fehler aufgetreten.

ErrGeneric Interner Fehler aufgetreten.

Mögliche Laufzeitfehler:

ErrHdlInvalid Einer der übergebenen Deskriptoren (obj) oder (handle5) ist ungültig.

ErrMemExhausted Nicht genug Speicher vorhanden.

ErrValueInvalid Im Kompressionsformat (int1) oder Kompressionsstufe (int2) wurde ein ungültiger Wert angegeben.

Eine der Längen- oder Positionsangaben (int3), (int4) oder (int6) ist ungültig.
Der komprimierte Inhalt kann im ungünstigsten Fall länger werden als der unkomprimierte Inhalt. Übersteigt die Länge den zu Verfügung stehenden Platz des Ziel-Memory-Objektes wird der Laufzeitfehler ebenfalls generiert.

Kontakt

obj -> MemCopy(int1, int2,
int3[, handle4])



Speicherbereich kopieren

obj Deskriptor des
 Memory-Objekts
 Position des
int1 ersten Bytes
 Anzahl der zu
int2 kopierenden
 Bytes
int3 Zielposition
 Deskriptor des
handle4 Ziel-Objekts
 Verwandte
Siehe Befehle

Dieser Befehl kopiert einen ausgewählten Speicherbereich des Memory-Objekts (obj) an eine andere Stelle des Speichers oder in ein anderes Memory-Objekt (handle4). Der Befehl kopiert (int2) Bytes ab der Position (int1) an die Zielposition (int3). Überlappende Bereiche werden beim Kopieren korrekt behandelt. Nach dem Kopieren wird der Wert der Eigenschaft Len des Zielobjekt gegebenenfalls erhöht.

Bei einer ungültigen Startposition (int1) oder einer zu großen Länge (int2) wird ein Laufzeitfehler erzeugt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergeben Deskriptor (obj) oder (handle4) ist ungültig.

ErrValueRange Die angegeben Startposition (int1) oder die Länge des zu kopierenden Bereiches (int2) sind zu groß.

Kontakt

obj -> MemDecrypt(int1, alpha2, alpha3[,
int4[, int5[, handle6[, int7[, handle8[,
handle9[, handle10]]]]]]]) : int



Speicherbereich entschlüsseln

obj Chiffrat / Klartext (Memory-Objekt)
int1 Optionen (siehe Text)
alpha2 Schlüssel
alpha3 Initialisierungsvektor
int4 Chiffratposition
int5 Chiffratlänge
handle6 Klartext (Memory-Objekt)
int7 Klartextposition
handle8 Schlüssel (Memory-Objekt)
handle9 Initialisierungsvektor (Memory-Objekt)
handle10 Authentifikations-Tag für GCM
(Memory-Objekt)

Fehlerwert

ErrOk Erfolg

ErrMemKeyInvalid Schlüssel ist

ungültig

ErrMemKeyLength Schlüssel ist zu kurz

oder zu lang für den

Algorithmus

Resultat int ErrMemIVInvalid Initialisierungsvektor
ist ungültig
ErrMemIVLength Initialisierungsvektor
ist zu kurz oder zu
lang für den
Algorithmus
ErrMemDecrypt Entschlüsselung ist
fehlgeschlagen

Siehe Verwandte Befehle, MemEncrypt(), Blog

Dieser Befehl entschlüsselt den Inhalt des Memory-Objektes (obj). Es müssen der gleiche Schlüssel und Initialisierungsvektor angegeben werden, der auch beim Verschlüsseln verwendet wurde.

Folgende Optionen (int1) sind möglich:

- Verschlüsselungsalgorithmus
 - MemCipherAES128 128-Bit-Verschlüsselung mit AES
 - MemCipherAES192 192-Bit-Verschlüsselung mit AES
 - MemCipherAES256 256-Bit-Verschlüsselung mit AES
 - MemCipherRSA Verschlüsselung mit RSA
- Verschlüsselungsmodus (nicht in Verbindung mit MemCipherRSA)
 - MemCipherModeCBC Betriebsmodus Cipher-block chaining MemCipherModeOFB
 - Betriebsmodus Output feedback

Kontakt

MemCipherModeCTR Betriebsmodus Counter Mode

MemCipherModeGCM Betriebsmodus Galois/Counter Mode

- Kodierung des Schlüssels

MemKeyHex Schlüssel liegt als hexadezimal kodierte Zeichenkette vor (alpha2)

MemKeyBase64 Schlüssel liegt Base64-kodierte Zeichenkette vor (alpha2)

MemKeyMem Schlüssel liegt binär vor (handle8)

- Typ des asymmetrischen Schlüssels (nur in Verbindung mit

MemCipherRSA)

MemKeyAsymPrivate Der Schlüssel (alpha2) / (handle8) ist ein privater asymmetrischer Schlüssel

MemKeyAsymPublic Der Schlüssel (alpha2) / (handle8) ist ein öffentlicher
asymmetrischer Schlüssel

- Kodierung des Initialisierungsvektors (nicht in Verbindung mit MemCipherRSA)

MemIVHex Initialisierungsvektor liegt als hexadezimal kodierte
Zeichenkette vor (alpha3)

MemIVBase64 Initialisierungsvektor liegt als Base64-kodierte Zeichenkette vor (alpha3)

MemIVMem Initialisierungsvektor liegt binär vor (handle9)

- Padding-Optionen

MemCipherNoPadding Kein Padding voraussetzen (nicht in Verbindung mit
MemCipherRSA)

MemPaddingRSAPkcs1 PKCS#1 Padding voraussetzen (nur in Verbindung mit
MemCipherRSA). Dieser Modus ist weit verbreitet.

MemPaddingRSAOaep OAEP Padding voraussetzen (nur in Verbindung mit
MemCipherRSA und MemKeyAsymPublic). Dieser Modus
sollte für neue Projekte verwendet werden.

Der Parameter (int1) setzt sich aus der Kombination je einer Konstanten der Bereiche Verschlüsselungsalgorithmus, Verschlüsselungsmodus, Kodierung des Schlüssels und Kodierung des Initialisierungsvektors zusammen. Diese können zusätzlich mit den Padding-Optionen kombiniert werden.



Wird die Option MemCipherNoPadding verwendet, muss die zu entschlüsselnde Nachricht 16 Byte lang oder ein Vielfaches davon sein.

Bei der Entschlüsselung mit MemCipherRSA muss ein Padding (MemPaddingRSA...) angegeben werden.

Der Schlüssel wird in (alpha2) bzw. (handle8) angegeben. Je nach Verschlüsselungsalgorithmus muss der Schlüssel eine bestimmte Länge haben:

Algorithmus Schlüssellänge (Rohdaten)

MemCipherAES128 128 Bit (16 Byte)

MemCipherAES192 192 Bit (24 Byte)

MemCipherAES256 256 Bit (32 Byte)

MemCipherRSA Die Verschlüsselungsstärke hängt von der Schlüssellänge ab.

Kontakt

 Im Gegensatz zu _MemCipherAES..., wo der Schlüssel aus zufälligen Zeichen besteht, muss bei _MemCipherRSA ein gültiger Schlüssel eines Schlüsselpaares angegeben werden. Dieses Schlüsselpaar kann beispielsweise mit MemGenKeyPair() oder OpenSSL erzeugt werden. Die maximale Länge der verschlüsselbaren Zeichenkette hängt hierbei vom gewählten Padding und der Bitlänge des Schlüssels ab.

Der asymmetrische Schlüssel muss im PKCS #1- oder im X.509-Format vorliegen.
Die Begin- und Endmarkierung dürfen nicht enthalten sein.

 Die Ver- und Entschlüsselung mit RSA nimmt wesentlich mehr Zeit in Anspruch als mit AES.
Um große Datenmengen zu verschlüsseln sollte ein zufälliger AES-Schlüssel generiert werden, der mit RSA verschlüsselt wird. Der eigentliche Klartext wird mit AES verschlüsselt.

Der Initialisierungsvektor wird in (alpha3) bzw. (handle9) angegeben. Die Länge vom Initialisierungsvektor ist abhängig von der Blocklänge des Verschlüsselungsalgorithmus. Bei AES sind dies 128 Bit (16 Byte). Der Initialisierungsvektor muss nicht geheim gehalten werden. Bei _MemCipherRSA wird der Initialisierungsvektor nicht verwendet.

In (int4) und (int5) wird die Position und die Länge der zu entschlüsselnden Daten in (obj) angegeben.

In (handle6) kann ein Ziel-Memory-Objekt angegeben werden, in dass der Klartext gespeichert wird. Ist dieses nicht angegeben, erfolgt die Speicherung in (obj).

Die Zielposition des Klartextes in (handle6) bzw. (obj) kann im Argument (int7) angegeben. Ist das Argument nicht gesetzt, wird an den Anfang des Memory-Objektes geschrieben.

In (handle10) muss ein Memory-Objekt angegeben werden, wenn als Option (int1) MemCipherModeGCM angegeben ist. In dieses Memory-Objekt muss zuvor das Authentifikations-Tag geschrieben, welches bei MemEncrypt() ermittelt wurde. Dieses Tag ist 16 Byte lang.

Mögliche Laufzeitfehler:

<u>ErrHdIInvalid</u>	Einer der übergebenen Deskriptoren (obj), (handle6), (handle8), (handle9) oder (handle10) ist ungültig.
<u>ErrMemExhausted</u>	Nicht genug Speicher vorhanden.
	Bei der Option <u>MemKeyMem</u> wurde kein Schlüssel (handle8), bei <u>MemIVMem</u> kein Initialisierungsvektor (handle9) oder bei <u>MemCipherModeGCM</u> kein Authentifikations-Tag (handle10) angegeben.
<u>ErrValueInvalid</u>	In den Optionen (int1) wurde eine ungültige oder unvollständige Kombination angegeben.
<u>ErrValueRange</u>	Eine der Längen- oder Positionsangaben (int4), (int5) oder (int7) ist ungültig.

Kontakt

obj -> MemEncrypt(int1, alpha2, alpha3[,
int4[, int5[, handle6[, int7[, handle8[,
handle9[, handle10]]]]]]]) : int



Speicherbereich verschlüsseln

obj Klartext / Chiffrat (Memory-Objekt)

int1 Optionen (siehe Text)

alpha2 Schlüssel

alpha3 Initialisierungsvektor

int4 Klartextposition

int5 Klartextlänge

handle6 Chiffrat (Memory-Objekt)

int7 Chiffratposition

handle8 Schlüssel (Memory-Objekt)

handle9 Initialisierungsvektor (Memory-Objekt)

Authentifikations-Tag für GCM

handle10 (Memory-Objekt)

Fehlerwert

ErrOk

Erfolg

ErrMemKeyInvalid Schlüssel ist

ungültig oder der
falsche Schlüsseltyp
bei asymmetrischen
Schlüsseln wurde
angegeben

Resultat int ErrMemKeyLength Schlüssel ist zu kurz
oder zu lang für den

Algorithmus

Initialisierungsvektor

ist ungültig

Initialisierungsvektor

ist zu kurz oder zu

lang für den

Algorithmus

ErrMemIVInvalid

ErrMemIVLength

Siehe Verwandte Befehle, MemDecrypt(), Blog
Dieser Befehl verschlüsselt den Inhalt des Memory-Objektes (obj).

Folgende Optionen (int1) sind möglich:

- Verschlüsselungsalgorithmen

MemCipherAES128 128-Bit-Verschlüsselung mit AES

MemCipherAES192 192-Bit-Verschlüsselung mit AES

MemCipherAES256 256-Bit-Verschlüsselung mit AES

MemCipherRSA Verschlüsselung mit RSA

- Verschlüsselungsmodus (nicht in Verbindung mit MemCipherRSA)

MemCipherModeCBC Betriebsmodus Cipher-block chaining MemCipherModeOFB

Betriebsmodus Output feedback

Kontakt

MemCipherModeCTR Betriebsmodus Counter Mode

MemCipherModeGCM Betriebsmodus Galois/Counter Mode

- Kodierung des Schlüssels

MemKeyHex Schlüssel liegt als hexadezimal kodierte Zeichenkette vor (alpha2)

MemKeyBase64 Schlüssel liegt Base64-kodierte Zeichenkette vor (alpha2)

MemKeyMem Schlüssel liegt binär vor (handle8)

- Typ des asymmetrischen Schlüssels (nur in Verbindung mit

MemCipherRSA)

MemKeyAsymPrivate Der Schlüssel (alpha2) / (handle8) ist ein privater asymmetrischer Schlüssel



Bei Verwendung des privaten Schlüssels erzeugt der gleiche Klartext immer das gleiche Chiffraut

MemKeyAsymPublic Der Schlüssel (alpha2) / (handle8) ist ein öffentlicher asymmetrischer Schlüssel

- Kodierung des Initialisierungsvektors (nicht in Verbindung mit MemCipherRSA)

MemIVHex Initialisierungsvektor liegt als hexadezimal kodierte

Zeichenkette vor (alpha3)

MemIVBase64 Initialisierungsvektor liegt als Base64-kodierte Zeichenkette vor (alpha3)

MemIVMem Initialisierungsvektor liegt binär vor (handle9)

- Padding-Optionen

MemCipherNoPadding Kein Padding anwenden (nicht in Verbindung mit MemCipherRSA)

MemPaddingRSAPkcs1 PKCS#1 Padding anwenden (nur in Verbindung mit MemCipherRSA). Dieser Modus ist weit verbreitet.

MemPaddingRSAOaep OAEP Padding anwenden (nur in Verbindung mit

MemCipherRSA und MemKeyAsymPublic). Dieser Modus sollte für neue Projekte verwendet werden.

Der Parameter (int1) setzt sich aus der Kombination je einer Konstanten der Bereiche Verschlüsselungsalgorithmus, Verschlüsselungsmodus, Kodierung des Schlüssels und Kodierung des Initialisierungsvektors zusammen. Diese können zusätzlich mit den Padding-Optionen kombiniert werden.



Wird die Option MemCipherNoPadding verwendet, muss die zu verschlüsselnde Nachricht 16 Byte lang oder ein Vielfaches davon sein.

Bei der Verschlüsselung mit MemCipherRSA muss ein Padding (MemPaddingRSA...) angegeben werden. Das Padding füllt den Rest der zu verschlüsselnden Nachricht auf die Schlüsselbitlänge auf. Für das Padding werden mindestens die folgende Anzahl an Zeichen benötigt:

MemPaddingRSAPkcs1 11 Zeichen

MemPaddingRSAOaep 42 Zeichen

Kontakt

 Auch wenn der zu verschlüsselnde Klartext im Memory-Objekt nur aus darstellbaren Zeichen besteht, kann das Chifferrat aus nichtdarstellbaren Zeichen (binären Daten) bestehen.

Der Schlüssel wird in (alpha2) bzw. (handle8) angegeben. Je nach Verschlüsselungsalgorithmus muss der Schlüssel eine bestimmte Länge haben:

<u>Algorithmus</u>	<u>Schlüssellänge (Rohdaten)</u>
<u>MemCipherAES128</u>	128 Bit (16 Byte)
<u>MemCipherAES192</u>	192 Bit (24 Byte)
<u>MemCipherAES256</u>	256 Bit (32 Byte)
<u>MemCipherRSA</u>	Die Verschlüsselungsstärke

i Im Gegensatz zu _MemCipherAES..., wo der Schlüssel aus zufälligen Zeichen besteht, muss bei _MemCipherRSA ein gültiger Schlüssel eines Schlüsselpaares angegeben werden. Dieses Schlüsselpaar kann beispielsweise mit MemGenKeyPair() oder OpenSSL erzeugt werden. Die maximale Länge der verschlüsselbaren Zeichenkette hängt hierbei vom gewählten Padding und der Bitlänge des Schlüssels ab.

**Der asymmetrische Schlüssel muss im PKCS #1- oder im X.509-Format vorliegen.
Die Begin- und Endmarkierung dürfen nicht enthalten sein.**

 Die Ver- und Entschlüsselung mit RSA nimmt wesentlich mehr Zeit in Anspruch als mit AES. Um große Datenmengen zu verschlüsseln sollte ein zufälliger AES-Schlüssel generiert werden, der mit RSA verschlüsselt wird. Der eigentliche Klartext wird mit AES verschlüsselt.

Der Initialisierungsvektor wird in (alpha3) bzw. (handle9) angegeben. Dieser dient dazu, dass die gleiche Nachricht mit dem gleichen Schlüssel nicht das gleiche Ergebnis liefert. Der Initialisierungsvektor sollte zufällig generiert und nicht mehrfach genutzt werden. Die Länge vom Initialisierungsvektor ist abhängig von der Blocklänge des Verschlüsselungsalgorithmus. Bei AES sind dies 128 Bit (16 Byte). Der Initialisierungsvektor muss nicht geheim gehalten werden. Bei MemCipherRSA wird der Initialisierungsvektor nicht verwendet.

In (int4) und (int5) wird die Position und die Lnge der zu verschlsselnden Daten in (obj) angegeben.

In (handle6) kann ein Ziel-Memory-Objekt angegeben werden, in dass das Chiffrat gespeichert wird. Ist dieses nicht angegeben, erfolgt die Speicherung in (obj).

Die Zielposition des Chiffrats in (handle6) bzw. (obj) kann im Argument (int7) angegeben werden. Ist das Argument nicht gesetzt, wird an den Anfang des Memory-Objektes geschrieben.

In (handle10) muss ein Memory-Objekt angegeben werden, wenn als Option (int1) MemCipherModeGCM angegeben ist. In dieses Memory-Objekt wird das Authentifikations-Tag geschrieben, welches bei MemDecrypt() zur Authentifizierung der verschlüsselten Nachricht angegeben werden muss. Dieses Tag ist 16 Byte lang.

Mögliche Laufzeitfehler:

ErrHdlInvalid

Kontakt

Einer der übergebenen Deskriptoren (obj), (handle6), (handle8), (handle9) oder (handle10) ist ungültig.

ErrMemExhausted Nicht genug Speicher vorhanden.

Bei der Option MemKeyMem wurde kein Schlüssel (handle8), bei MemIVMem kein Initialisierungsvektor (handle9) oder bei MemCipherModeGCM kein Memory-Objekt für das Authentifikations-Tag (handle10) angegeben.

ErrStringOverflow

Der Plaintext (obj) ist bei der RSA-Verschlüsselung zusammen mit dem Padding länger als der Schlüssel (alpha2) bzw. (handle8).

ErrValueInvalid

In den Optionen (int1) wurde eine ungültige oder unvollständige Kombination angegeben.
Eine der Längen- oder Positionsangaben (int4), (int5) oder (int7) ist ungültig.

ErrValueRange

Kontakt

obj ->
MemFindByte(int1,|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||<img alt="Icon of a question mark" data-bbox="

Kontakt

obj -> MemFindStr(int1,
int2, alpha3[, int4]) : int



Zeichenkette suchen

obj Deskriptor des Memory-Objekts

int1 Startposition

Anzahl der zu durchsuchenden

int2 Bytes

alpha3 zu suchende Zeichenkette

Optionen (optional)

StrCaseIgnore Keine Unterscheidung
zwischen

Groß-/Kleinschreibung

int4 StrFindReverse Suche vom Ende der
Zeichenkette bis zur

Startposition

StrFindToken Begriffsorientierte
Suche

Resultat int Position des ersten Zeichens

Siehe Verwandte Befehle, MemFindByte()

Diese Funktion sucht im Speicherbereich des Memory-Objekts (obj) ab der Position (int1) in (int2) Bytes nach dem ersten Vorkommen der Zeichenkette (alpha3). Das Resultat ist 0, wenn der Wert nicht gefunden wurde. Ansonsten wird die Position des ersten gefundenen Werts zurückgeliefert.

Soll das gesamte Objekt nach einer Zeichenkette durchsucht werden, kann in (int2) die Datenmenge des Memory-Objekts (siehe Eigenschaft Len) angegeben werden. Soll zu Beginn des Objekts einige Bytes übersprungen werden, müssen diese von der Anzahl der zu durchsuchenden Bytes abgezogen werden.

Folgende Optionen (int4) sind zulässig:

- StrCaseIgnore

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

- StrFindReverse

Die Suche beginnt am Ende der Zeichenkette und endet an der Position (int3).

- StrFindToken

Die Suchergebnisse beschränken sich auf ganze Wörter.

Beispiel:

```
// Suche ab Anfang des ObjektstStartByte # 1;tSearchBytes # tMemory->spLen;tRes # tMemory->MemFin
```

Folgendes Beispiel ermittelt die Anzahl der Zeilenwechsel in einem Memory-Objekt:

```
for tPos # tMemory->MemFindStr(1, tMemory->spLen, StrChar(13) + StrChar(10));loop tPos # tMemo
```

Kontakt

Abhängig vom verwendeten Zeichensatz entspricht ein Byte nicht unbedingt einem Zeichen. Bei der Verwendung zum Beispiel von UTF-8 ist ein Zeichen zwischen einem und vier Bytes lang. Die Suche kann nur am Beginn eines Zeichens gestartet werden, sonst wird der Laufzeitfehler ErrValueRange erzeugt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

Die in (int1) oder (int2) übergebenen Werte liegen außerhalb des ErrValueRange zulässigen Bereichs oder die Suche beginnt nicht an einer Zeichengrenze.

Kontakt

obj -> MemFree()  Memory-

Objekt entfernen

Deskriptor eines
obj

Memory-Objekts

Verwandte

Siehe Befehle,

MemAllocate()

Mit dieser Funktion wird ein mittels MemAllocate() angelegtes Memory-Objekt wieder entfernt und der Speicherbereich freigegeben. In (obj) wird der Deskriptor des Objekts übergeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergeben Deskriptor ist ungültig.

Kontakt

obj -> MemGenKeyPair(handle1, int2) :

int

Asymmetrisches Schlüsselpaar erzeugen

**Privater
asymmetrischer**

Schlüssel (Memory-Objekt)

**Öffentlicher
asymmetrischer**

Schlüssel

Schlüssellänge in

int2
Bit

Bit Fehlerwerk

Resultat int

— ErrC

Verwa
Refakla

Siehe Blechle,

MemEncrypt()

MemDecrypt()

Dieser Befehl erzeugt ein RSA-Schlüsselpaar für asymmetrische Verschlüsselung (siehe [MemEncrypt\(\)](#) und [MemDecrypt\(\)](#) mit [MemCipherRSA](#)) bzw. Signierung (siehe [MemSign\(\)](#) und [MemVerify\(\)](#) mit [MemSignRSA](#)).



Die Bitlnge sollte eine Zweierpotenz sein >= 2048 sein (z. B. 2048, 4096)

Je höher die Schlüssellänge ist, desto länger dauert die Ver- bzw. Entschlüsselung.

Dennoch sollten Schlüssellängen ≤ 1024 Bit nicht mehr verwendet werden.

Beispiel:

```
tMemPrivate->MemGenKeyPair(tMemPublic, 2048);
```

Mögliche Laufzeitfehler:

— 1 —

Einer der übergebenen Deskriptoren (obj) oder (handle1) ist ungültig.

ErrMemExhausted Nicht genug Speicher vorhanden.

ErrValueInvalid Es wurde eine ungültige Schlüssellänge (int2) angegeben.

Kontakt

obj -> MemHash(int1[, int2[,
int3]]) : alpha



Authentifizierungs-Code ermitteln

obj Deskriptor des Memory-Objekts

Hash-Verfahren und Kodierungen

MemHashMD5 MD5-Hash
(Message-Digest
Algorithm 5)

MemHashRMD160 RIPEMD

160-Hash (RACE
Integrity
Primitives
Evaluation
Message Digest)

int1 MemHashSHA1 SHA-1 Hash
(Secure Hash
Algorithm)

MemHashSHA256 SHA-256 Hash

MemHashSHA384 SHA-384 Hash

MemHashSHA512 SHA-512 Hash

MemResultHex Hexadezimales
Ergebnis

MemResultBase64 Base64-kodiertes

Ergebnis

int2 Startposition für die Berechnung
(optional)

int3 Länge (optional)

Resultat alpha Hash-Wert

Siehe Verwandte Befehle, MemHMAC()

Dieser Befehl bildet für eine Nachricht einen Authentifizierungs-Code. Der Code wird aus dem Inhalt des übergebenen Memory-Objekts (obj) und dem Verfahren (int1) errechnet. Im Parameter (int1) wird auch die Kodierung des Ergebnisses angegeben. Folgende Konstanten können übergeben werden:

- Hash-Verfahren

MemHashMD5 MD5-Hash (Message-Digest Algorithm 5)

MemHashRMD160 RIPEMD 160-Hash (RACE Integrity Primitives Evaluation)

Message Digest)

MemHashSHA1 SHA-1 Hash (Secure Hash Algorithm)

MemHashSHA256 SHA-256 Hash

MemHashSHA384 SHA-386 Hash

MemHashSHA512 SHA-512 Hash

- Kodierung des Ergebnisses

MemResultHex Hexadezimales Ergebnis

MemResultBase64 Base64-kodiertes Ergebnis

Kontakt



Der Parameter (int1) setzt sich aus der Kombination je einer Konstanten aus den Bereichen Verfahren und Ergebnis-Kodierung zusammen.

Beispiel:

```
tHashValue # tMem->MemHash(_MemHashSHA256 | _MemResultHex);
```

Der Hash-Wert kann auch aus einem Teil des Memory-Objekts gebildet werden. Dazu wird die Startposition (int2) und die zu berücksichtigende Länge (int3) angegeben. Wird eine ungültigen Startposition (int2) oder eine zu große Länge (int3) angegeben, wird ein Laufzeitfehler erzeugt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebenen Deskriptor (obj) ist ungültig.

Als Hash- und Kodierungsverfahren (int1) wurde eine ungültige /

ErrValueInvalid unvollständige Kombination angegeben. Aus jeder Gruppe muss eine Konstante angegeben werden.

ErrValueRange Die angegeben Startposition (int2) oder die Länge des Bereiches (int3) ist zu groß.

`obj -> MemHMAC(int1, alpha2[, int3[,
int4[, handle5]]]) : alpha`



Authentifizierungs-Code ermitteln

`obj` Deskriptor des Memory-Objekts

Hash-Verfahren und Kodierungen

MemHashMD5 MD5-Hash
(Message-Digest)
Algorithm 5

MemHashRMD160 RIPEMD 160-Hash (RACE
Integrity
Primitives
Evaluation

MemHashSHA1 Message Digest)
SHA-1 Hash
(Secure Hash
Algorithm)

MemHashSHA256 SHA-256 Hash

MemHashSHA384 SHA-384 Hash

`int1` MemHashSHA512 SHA-512 Hash

MemKeyHex Schlüssel liegt

MemKeyBase64 hexadezimal vor
Schlüssel liegt
Base64-kodiert

MemKeyMem vor
Schlüssel liegt
als
Memory-Objekt
(handle5) vor

MemResultHex Hexadezimales
Ergebnis

MemResultBase64 Base64-kodiertes
Ergebnis

`alpha2` Schlüssel des Hash-Verfahrens

`int3` Startposition für die Berechnung
(optional)

`int4` Länge (optional)

`handle5` Memory-Objekt mit Schlüssel des
Hash-Verfahrens (optional)

Resultat alpha Authentifizierungscode

Siehe Verwandte Befehle, MemHash()

Dieser Befehl bildet für eine Nachricht einen Authentifizierungs-Code. Der Code wird aus dem Inhalt des übergebenen Memory-Objekts (obj), dem Verfahren (int1) und dem Schlüssel (alpha2) errechnet.

Der Schlüssel kann alternativ in einem Memory-Objekt vorliegen, welches in (alpha5) angegeben wird.

Dazu muss in (int1) MemKeyMem

Kontakt

angegeben sein. Es werden maximal die ersten 1024 Byte des Schlüssels aus dem Memory-Objekt berücksichtigt.

Im Parameter (int1) wird auch die Kodierung des Ergebnisses angegeben. Folgende Konstanten können übergeben werden:

- Hash-Verfahren

MemHashMD5 MD5-Hash (Message-Digest Algorithm 5)

MemHashRMD160 RIPEMD 160-Hash (RACE Integrity Primitives Evaluation

Message Digest)

MemHashSHA1 SHA-1 Hash (Secure Hash Algorithm)

MemHashSHA256 SHA-256 Hash

MemHashSHA384 SHA-384 Hash

MemHashSHA512 SHA-512 Hash

- Kodierung des Schlüssels

MemKeyHex Schlüssel liegt hexadezimal vor

MemKeyBase64 Schlüssel liegt Base64-kodiert vor

MemKeyMem Schlüssel liegt Memory-Objekt vor

- Kodierung des Ergebnisses

MemResultHex Hexadezimales Ergebnis

MemResultBase64 Base64-kodiertes Ergebnis



Der Parameter (int1) setzt sich aus der Kombination je einer Konstanten aus den Bereichen Verfahren, Schlüssel-Kodierung und Ergebnis-Kodierung zusammen.

Beispiele:

```
tHashValue # tMem->MemHMAC(_MemHashSHA256 | _MemKeyBase64 | _MemResultHex, aKeyHMAC);tHashValue #
```

Der Hash-Wert kann auch aus einem Teil des Memory-Objekts gebildet werden. Dazu wird die Startposition (int3) und die zu berücksichtigende Länge (int4) angegeben. Wird eine ungültigen Startposition (int3) oder eine zu große Länge (int4) angegeben, wird ein Laufzeitfehler erzeugt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebenen Deskriptor (obj) oder (handle5) ist ungültig.

Als Hash- und Kodierungsverfahren (int1) wurde eine ungültige /

ErrValueInvalid unvollständige Kombination angegeben. Aus jeder Gruppe muss eine Konstante angegeben werden.

ErrValueRange Die angegeben Startposition (int3) oder die Länge des Bereiches (int4) ist zu groß.

Kontakt

obj ->
MemReadByte(int1)        
: int
Bytewert lesen
 Deskriptor des
obj
 Memory-Objekts
 Position des zu
int1
 lesenden Bytes
Resultat int gelesenes Byte —
 Verwandte Befehle,
Siehe [MemWriteByte\(\)](#),
 [MemReadStr\(\)](#)

Der Befehl ermittelt den Bytewert im Speicherbereich des Memory-Objekts (obj) an der Position (int1).

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

ErrValueRange Die in (int1) angegebene Position ist nicht gültig.

Kontakt

obj -> _____
MemReadStr(int1,
int2[, int3]) : alpha
Zeichenkette lesen
 Deskriptor des
 obj Memory-Objekts
int1 Erstes zu lesendes Byte
 Anzahl, der zu lesenden
int2 Bytes
 Zielzeichensatz
int3 (optional)
 gelesene



Resultat alpha Zeichenkette
Verwandte Befehle,
Siehe MemWriteStr(),
 MemReadByte()

Dieser Befehl liest aus dem Speicherbereich des Memory-Objekts (obj) ab der Position (int1) eine Menge von (int2) Bytes und überträgt sie in den Resultatwert. Dabei wird eine Zeichensatzwandlung durchgeführt, wenn in (int3) ein Zielzeichensatz (siehe Charset) angegeben wird. Als Quellzeichensatz wird die Eigenschaft Charset des Memory-Objekts verwendet. Es kann zwischen allen unterstützten Zeichensätzen konvertiert werden. Dabei werden Zeichen, die nicht im Zielzeichensatz darstellbar sind, durch ein Fragezeichen ersetzt.

-  In einigen Zeichensätzen werden mehrere Bytes zur Repräsentation eines Zeichens verwendet.
Wird eine Anzahl von Bytes angegeben, die nicht auf eine Zeichengrenze fällt, kann das letzte Zeichen nicht konvertiert werden.
-  Da Zeichenketten null-terminiert sind, kann die Länge der resultierenden Zeichenkette kleiner als die zu lesende Länge (int2) sein, sofern das Memory-Objekt im ausgelesenen Bereich ein Null-Byte enthält.

Mögliche Laufzeitfehler:

- ErrHdIInvalid Der in (obj) übergebene Descriptor ist ungültig.
- ErrValueRange Die Position (int1) oder die Länge (int2) liegen außerhalb des zulässigen Bereichs.
- ErrValueInvalid In (int3) wurde ein undefinierter Zeichensatz angegeben.

Kontakt

obj -> MemResize(**int1**) : int  Größe des
Memory-Objekts verändern

Deskriptor des
obj

int1 Neue Größe des
Objekts

Resultat int Fehlerwert 
Verwandte

Siehe Befehle, Size

Mit dieser Funktion wird die Größe des Speicherbereichs des Memory-Objekts (obj) verändert. In (int1) wird die neue Größe in Bytes angegeben. Die neue Größe muss dabei im Bereich von 1 Byte bis 512 MB (bei 32-Bit-Prozessen) bzw. 2 GB (bei 64-Bit-Prozessen) liegen. Bei einer Verkleinerung werden eventuell überstehende Daten verworfen.

Konnte das Objekt verändert werden, wird `ErrOk` zurückgegeben. Steht keine ausreichende Menge an Hauptspeicher zur Verfügung, ist das Resultat `ErrOutOfMemory`.

Folgende Fehlerwerte sind möglich:

_ErrValueInvalid Als Größe (int1) wurde ein Wert ≤ 0 angegeben.

ErrLimitExceeded Bei einem 32-Bit-Prozess wurde eine Größe (int1) von mehr als 512 MB angegeben.

ErrOutOfMemory Der Speicher konnte nicht angefordert werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Descriptor ist ungültig.

Kontakt

obj -> MemSign(int1,
alpha2, var alpha3[,
int4[, int5]]) : int
Signatur erstellen



obj Nachricht (Memory-Objekt)
Optionen

MemSignRSA RSA-Signatur
erstellen

MemSignDSA DSA-Signatur
erstellen

MemHashMD5 MD5-Hash
verwenden

MemHashRMD160 RIPEMD-160-Hash

verwenden

MemHashSHA1 SHA-1-Hash
verwenden

MemHashSHA256 SHA-256-Hash
verwenden

MemHashSHA384 SHA-384-Hash
verwenden

int1
MemHashSHA512 SHA-512-Hash
verwenden

MemKeyHex Schlüssel liegt als
hexadezimal
kodierte

MemKeyBase64 Zeichenkette vor
Schlüssel liegt
Base64-kodierte
Zeichenkette vor

MemResultHex Signatur soll
Hex-Codiert sein

MemResultBase64 Signatur soll
Base64-Codiert
sein

alpha2 Schlüssel

var Signatur

alpha3

int4 Nachrichtenposition (optional)

int5 Nachrichtenlänge (optional)

Resultat int Fehlerwert

ErrOk Erfolg

ErrMemKeyInvalid Schlüssel
ungültig

Siehe

ErrMemKeyLength Schlüssel

zu kurz für
Nachricht

Verwandte Befehle, MemVerify(),

Blog

Dieser Befehl signiert eine Nachricht nach dem in (int1) angegebenen Verfahren. Die Signatur wird aus dem Inhalt des übergebenen Memory-Objektes (obj), dem Verfahren (int1) und dem privaten Schlüssel (alpha2) gebildet. Im Parameter (int1) wird ebenfalls der zu verwendete Hash-Algorithmus, die Codierung des privaten Schlüssels und die Codierung der Signatur angegeben.

Folgende Optionen (int1) sind möglich:

- Typ der Signatur

MemSignRSA RSA-Signatur erstellen
MemSignDSA DSA-Signatur erstellen

- Hash-Verfahren

MemHashMD5 MD5-Hash verwenden
MemHashRMD160 RIPEMD-160-Hash verwenden
MemHashSHA1 SHA-1-Hash verwenden
MemHashSHA256 SHA-256-Hash verwenden
MemHashSHA384 SHA-384-Hash verwenden
MemHashSHA512 SHA-512-Hash verwenden

- Kodierung des Schlüssels

MemKeyHex Schlüssel liegt als hexadezimal kodierte Zeichenkette vor
MemKeyBase64 Schlüssel liegt Base64-kodierte Zeichenkette vor

- Kodierung des Ergebnisses

MemResultHex Signatur soll Hex-Codiert sein
MemResultBase64 Signatur soll Base64-Codiert sein

Der Parameter (int1) setzt sich aus der Kombination je einer Konstanten der Bereiche zusammen.

Die ermittelte Signatur wird in (alpha3) zurückgegeben.

Der private Schlüssel (alpha2) muss im PKCS #1- (RSA oder DSA) oder im PKCS #8-Format (RSA oder DSA) vorliegen. Die Begin- und Endmarkierung dürfen nicht enthalten sein.

Ein RSA-Schlüsselpaar mit 2048 Bit Länge kann beispielsweise wie folgt angelegt werden:

```
tMemPrivate->MemGenKeyPair(tMemPublic, 2048);
```

Alternativ ist dies mit OpenSSL wie folgt möglich:

```
openssl genrsa -out rsa_priv_key.pem 2048openssl rsa -in rsa_priv_key.pem -pubout -out rsa_pub_ke
```

Kontakt

Mit der folgenden Funktion kann der Schlüssel eingelesen werden:

```
sub ReadKey( aKeyFile    : alpha(250); var aKey      : alpha; opt aPublic : logic; ): int; local
```

Optional kann mit den Argumenten (int4) die Position und (int5) die Länge der zu verwendenden Nachricht angegeben werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid

Der übergeben Deskriptor (obj) ist ungültig.

ErrMemExhausted

Nicht genug Speicher für kodierte Signatur (vor der Rückgabe) vorhanden.

ErrStringOverflow

Der Schlüssel (alpha2) ist länger als 8192 Zeichen oder die in (alpha3) übergebene Variable ist nicht lang genug zum Speichern der Signatur.

ErrValueInvalid

In den Optionen (int1) wurde eine ungültige oder unvollständige Kombination angegeben.

ErrValueRange

Die angegeben Startposition (int4) oder die Länge des Bereiches (int5) ist zu groß.

Kontakt

obj -> MemUncompress([int1[, int2[, handle3[, int4]]]]) : int Speicherbereich
dekompprimieren



Quelle / Ziel
obj (Memory-Objekt)
Quellposition
int1 (optional)
int2 Quelllänge (optional)
handle3 Ziel (Memory-Objekt,
optional)
int4 Zielposition (optional)
Resultat int Fehlerwert 
Verwandte Befehle,
Siehe MemCompress(),
FsiFileUncompress()

Dieser Befehl dekomprimiert den Inhalt des Memory-Objektes (obj).

Im Parameter (int1) kann die Quellposition angegeben werden. Ist dieser Wert nicht angegeben oder 0, werden die Daten ab Beginn des Memory-Objektes dekomprimiert.

Der Parameter (int2) gibt die zu dekomprimierende Länge an. Ist dieser Wert nicht angegeben, 0 oder MemDataLen wird der restliche Inhalt (nach der Quellposition) des Memory-Objektes dekomprimiert.

Optional kann im Parameter (handle3) ein Ziel-Memory-Objekt angegeben werden. Ist dieses nicht angegeben oder ist es identisch mit dem Quellobjekt (obj), wird in das Quellobjekt geschrieben. Hierbei werden alle Daten verworfen, die hinter dem letzten dekomprimierten Zeichen vorhanden waren.

Zusätzlich kann eine Zielposition (int4) angegeben werden, wenn nicht an den Anfang des Ziel-Objektes geschrieben werden soll. Alle vorhandenen Daten ab der Position werden überschrieben.

Beispiele

```
// Inhalt des Memory-Objektes tMemSrc in neues Memory-Objekt dekomprimieren tMemSrc->MemUncompress
```

Fehlerwerte

Folgende Fehlerwerte können von der Funktion zurückgegeben werden:

ErrOk Kein Fehler aufgetreten.

ErrData Komprimierte Daten sind inkonsistent oder Quellobjekt (obj) ist leer.

ErrGeneric Interner Fehler aufgetreten.

Mögliche Laufzeitfehler:

ErrHdIInvalid Einer der übergebenen Deskriptoren (obj) oder (handle3) ist ungültig.

ErrMemExhausted Nicht genug Speicher vorhanden.

Kontakt

ErrValueRange

**Eine der Längen- oder Positionsangaben (int1), (int2) oder (int4)
ist ungültig.**

Kontakt

obj -> MemVerify(int1,



alpha2, alpha3[, int4[,
int5]] : int

Signatur überprüfen

obj Nachricht (Memory-Objekt)

Optionen

MemSignRSA

RSA-Signatur
verifizieren

MemSignDSA

DSA-Signatur
verifizieren

MemHashMD5

MD5-Hash

verwenden

RIPEMD-160-Hash

verwenden

SHA-1-Hash

verwenden

SHA-256-Hash

verwenden

SHA-384-Hash

int1

MemHashSHA512

verwenden

verwenden

MemKeyHex

Schlüssel liegt als
hexadezimal
kodierte

Zeichenkette vor

MemKeyBase64

Schlüssel liegt
Base64-kodierte

Zeichenkette vor

MemSignatureHex

Signatur liegt

Hex-Codiert vor

MemSignatureBase64

Signatur liegt
Base64-Codiert

vor

alpha2 Schlüssel

alpha3 Signatur

int4 Nachrichtenposition (optional)

int5 Nachrichtenlänge (optional)

Resultat int Fehlerwert

ErrOk

Signatur
passt zur
Nachricht

ErrMemKeyInvalid Schlüssel

Kontakt

ungültig

ErrMemSgnInvalid Signatur

ungültig

ErrMemMsgVerify Signatur

passt nicht

zur

Nachricht

Siehe **Verwandte Befehle, MemSign()**,

Blog

Dieser Befehl prüft, ob die in (alpha3) angegebene Signatur zum öffentlichen Schlüssel (alpha2) und der im Memory-Objekt (obj) liegenden Nachricht passt. Im Parameter (int1) wird neben dem, verwendeten Verfahren, der Hash-Algorithmus sowie die Codierung des öffentlichen Schlüssels und die der Signatur angegeben.

Folgende Optionen (int1) sind möglich:

- Typ der Signatur

MemSignRSA RSA-Signatur verifizieren

MemSignDSA DSA-Signatur verifizieren

- Hash-Verfahren

MemHashMD5 MD5-Hash verwenden

MemHashRMD160 RIPEMD-160-Hash verwenden

MemHashSHA1 SHA-1-Hash verwenden

MemHashSHA256 SHA-256-Hash verwenden

MemHashSHA384 SHA-384-Hash verwenden

MemHashSHA512 SHA-512-Hash verwenden

- Kodierung des Schlüssels

MemKeyHex Schlüssel liegt als hexadezimal kodierte Zeichenkette vor

MemKeyBase64 Schlüssel liegt Base64-kodierte Zeichenkette vor

- Kodierung der Signatur

MemSignatureHex Signatur liegt Hex-Codiert vor

MemSignatureBase64 Signatur liegt Base64-Codiert vor

Der Parameter (int1) setzt sich aus der Kombination je einer Konstanten der Bereiche zusammen.

Der öffentliche Schlüssel (alpha2) muss im PKCS #1- (nur RSA) oder im X.509-Format (RSA oder DSA) vorliegen. Die Begin- und Endmarkierung dürfen nicht enthalten sein.

Optional kann mit den Argumenten (int4) die Position und (int5) die Länge der zu verwendenden Nachricht angegeben werden.

Über den Rückgabewert kann ermittelt werden, ob die Signatur zum öffentlichen Schlüssel (alpha2) und der Nachricht (obj) passt (**ErrOk**) oder nicht (**ErrMemMsgVerify**).

Kontakt

Mögliche Laufzeitfehler:

- | | |
|---------------------------------|---|
| <u>ErrHdlInvalid</u> | Der übergebenen Deskriptor (obj) ist ungültig. |
| <u>ErrStringOverflow</u> | Der Schlüssel (alpha2) ist länger als 8192 Zeichen. |
| <u>ErrValueInvalid</u> | In den Optionen (int1) wurde eine ungültige oder unvollständige Kombination angegeben. |
| <u>ErrValueRange</u> | Die angegeben Startposition (int4) oder die Länge des Bereiches (int5) ist zu groß. |

Kontakt

obj ->
MemWriteByte(int1, int2[, int3])
Bytewert schreiben
 Deskriptor des
 obj
 Memory-Objekts
int1 zu beschreibende
 Position
 zu schreibender
int2 Wert
 Anzahl der Bytes,
 die geschrieben
 werden sollen
 (optional)
int3



Siehe Befehle,

MemReadByte(),
MemWriteStr()

Mit diesem Befehl wird der Bytewert (int2) in den Speicherbereich des Memory-Objekts (obj) an die Position (int1) geschrieben. Durch die Angabe einer Länge (int3) kann ein kompletter Speicherabschnitt mit diesem Wert gefüllt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

ErrValueRange Die Position (int1) oder die Länge (int3) liegen außerhalb des zulässigen Bereichs.

Kontakt

obj ->
MemWriteStr(int1,
alpha2[, int3])
Zeichenkette schreiben
Deskriptor des
obj
Memory-Objekts
Position des
ersten zu
int1
schreibenden
Bytes
alpha2 zu schreibende
Zeichenkette
Zeichensatz der
int3 Zeichenkette
(optional)
Verwandte



Siehe Befehle,
MemReadStr(),
MemWriteByte()

Dieser Befehl schreibt die Daten des alphanumerischen Werts (alpha2) in den Speicherbereich des Memory-Objekts (obj) ab der Position (int1). Dabei wird eine Zeichensatzwandelung durchgeführt, wenn in (int3) ein Quellzeichensatz (siehe Charset) angegeben wird. Als Zielzeichensatz wird die Eigenschaft Charset des Memory-Objekts verwendet. Es kann zwischen allen unterstützten Zeichensätzen konvertiert werden. Dabei werden Zeichen, die nicht im Zielzeichensatz darstellbar sind, durch ein Fragezeichen ersetzt.

Mögliche Laufzeitfehler:

- ErrHdlInvalid Der in (obj) übergebene Desriptor ist ungültig.
- ErrValueRange Die Position (int1) liegt außerhalb des zulässigen Bereichs.
- ErrValueInvalid In (int3) wurde ein undefinierter Zeichensatz angegeben.

Kontakt

obj -> MsxReadMem(handle1, int2, int3) :



int

Nachrichtenkanal in Memory-Objekt lesen

Deskriptor eines

obj

Nachrichtenkanals

handle1 Deskriptor eines

Memory-Objekts

Zielposition im

int2

Memory-Objekt

Anzahl der zu

int3

lesenden Bytes

Resultat int Fehlerwert



Verwandte Befehle,

Siehe

[MsxWriteMem\(\)](#)

Dieser Befehl liest binäre Daten aus dem Nachrichtenkanal (obj) in das **Memory-Objekt** (handle1) ein.

Die Funktion entspricht damit [MsxRead\(MsxData, ...\)](#). Vor dem Aufruf von MsxReadMem() muss ein Nachrichtenelement bereits mit [MsxRead\(MsxItem, ...\)](#) geöffnet sein. In (int2) wird die Zielposition im Memory-Objekt und in (int3) die Datenlänge angegeben. Die Datenlänge muss identisch mit der beim Schreiben angegebenen Länge sein (siehe [MsxWriteMem\(\)](#)).

Das Resultat enthält den Fehlerwert oder [ErrOk](#), wenn kein Fehler aufgetreten ist.

Mögliche Laufzeitfehler:

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

[ErrValueRange](#) Der in (int2) oder (int3) übergebene Wert ist außerhalb des zulässigen Bereichs.

Kontakt

obj -> MsxWriteMem(handle1, int2, int3) : int |  Nachrichtenkanal
schreiben aus einem Memory-Objekt

obj Deskriptor eines Nachrichtenkanals

handle1 Deskriptor eines Memory-Objekts

int2 Startposition im Memory-Objekt

int3 Anzahl der zu schreibenden Bytes

Resultat int Anzahl der geschriebenen Bytes oder Fehlerwert

Siehe [Verwandte Befehle, MsxReadMem\(\)](#)

Dieser Befehl schreibt binäre Daten aus dem Memory-Objekt (handle1) in den Nachrichtenkanal (obj). Die Funktion entspricht damit MsxWrite(MsxData, ...). Vor dem Aufruf von

MsxWriteMem() muss ein Nachrichtenelement bereits mit MsxWrite(MsxItem, ...) geöffnet sein.

In (int2) wird die Startposition im Memory-Objekt und in (int3) die Datenlänge angegeben. Die Funktion erzeugt im Nachrichten-Stream ein binäres Feld mit der Länge (int3). Zum Einlesen der Daten dieses binären Felds muss exakt die gleiche Länge bei MsxReadMem() angegeben werden.

Das Resultat enthält den Fehlerwert oder ErrOk, wenn kein Fehler aufgetreten ist.

Mögliche Laufzeitfehler:

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

ErrValueRange Der in (int2) oder (int3) angegebene wert ist außerhalb des zulässigen Bereichs.

Kontakt

obj -> PdfTextExtractMem(handle1) : int 

Textinformationen einer PDF-Seite ermitteln

obj Deskriptor eines

PDF-Objekts

handle1 Deskriptor eines
Memory-Objekts

Resultat int Fehlerwert

Verwandte

Siehe Befehle,

PdfPageOpen()

Mit dieser Anweisung wird die Textinformation einer zuvor mit PdfPageOpen() geöffneten Seite eines PDF-Dokuments ermittelt und in das Memory-Objekt (handle1) übertragen.

Ist keine Seite geöffnet, gibt die Anweisung den Fehlerwert ErrPdfPageClosed zurück. Der ermittelte Text wird an das Memory-Objekt angehängt (siehe Len). Der Zeichensatz des Memory-Objekts wird entsprechend der Eigenschaft Charset berücksichtigt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Desktiptor ist ungültig.

Kontakt

obj -> SckReadMem(int1, handle2, int3,
int4) : int



Vom Socket in Memory-Objekt lesen

obj Socket-Deskriptor

Optionen

SckLine komplette Zeile lesen

int1 SckReadMax Lesen bis maximale Länge /
 Lesen aller empfangener
 Zeichen

handle2 Deskriptor des Memory-Objekts

int3 Startposition im Memory-Objekt

int4 Anzahl der zu lesenden Bytes

Resultat int Anzahl der gelesenen Bytes oder
 Fehlerwert

Siehe Verwandte Befehle, SckWriteMem()

Mit dieser Funktion werden Daten vom Socket (obj) gelesen. Durch die Angabe einer Option in (int1) kann die Anzahl der Zeichen auf eine Zeile oder auf die Menge der verfügbaren Daten begrenzt werden. In (handle2) muss der Deskriptor eines Memory-Objekts angegeben werden. In (int3) wird die Startposition und (int4) die Anzahl der maximal zu lesenden Bytes übergeben. Gegebenenfalls wird der Wert der Eigenschaft Len erhöht.

Das Resultat gibt die Anzahl der gelesenen Bytes zurück. Ist das Resultat negativ ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder (handle2) übergebene Deskriptor ist ungültig.

ErrMemExhausted Der Speicher konnte nicht angefordert werden.

ErrValueRange Der in (int3) oder (int4) übergebene Wert ist außerhalb des zulässigen
 Bereichs.

Kontakt

obj -> SckWriteMem(int1, handle2, int3, int4) :



int

Vom Memory-Objekt auf den Socket schreiben

obj Socket-Deskriptor

 Optionen

SckBuffered Daten puffern

int1 SckLine nach den Daten automatisch ein

 CR/LF senden

handle2 Deskriptor eines Memory-Objekts

int3 Startposition im Memory-Objekt

int4 Anzahl der zu schreibenden Bytes

Resultat int Anzahl geschriebener Bytes oder Fehlerwert Siehe

Verwandte Befehle, SckReadMem()

Mit dieser Funktion werden Daten auf den Socket (obj) geschrieben. Mit der Option SckLine in (int1) wird nach den Daten automatisch ein CR/LF gesendet. Mit der Option SckBuffered werden die Daten zunächst zwischengespeichert und entweder beim Erreichen der Puffergrenze (4 KB) oder durch den Aufruf von SckWrite() mit einer Länge von 0 geschrieben. Dadurch werden die Daten von mehreren SckWriteMem()-Aufrufen für das Versenden zusammengefasst.

In (handle2) muss der Deskriptor eines Memory-Objekts angegeben werden. In (int3) wird die Startposition und (int4) die Anzahl der zu schreibenden Bytes übergeben.

Das Resultat gibt die Anzahl der geschriebenen Bytes zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder (handle2) übergeben Deskriptor ist ungültig.

 Der in (int3) oder (int4) übergebene Wert ist außerhalb des

ErrValueRange zulässigen Bereichs.

Kontakt

obj -> WinRtfPicInsertMem(handle1[, int2[, int3]]) : int



Einfügen von Bildern in ein RtfEdit-Objekt

obj Deskriptor des RtfEdit-Objekts

handle1 Deskriptor des Memory-Objekts

int2 Seitennummer (optional)

Optionen (optional)

WinRtfPicModeSpeed Performantes

Einfügen

WinRtfPicModeQuality Qualitatives

Einfügen

int3 WinRtfPicModeAuto

Modus

abhängig von
der Farbtiefe
und Größe
des Bildes
auswählen

Resultat int Einfügeresultat (siehe Text)



Verwandte Befehle,

Siehe

WinRtfPicInsertName()

Diese Funktion fügt ein Bild aus einem Memory-Objekt (handle1) an der aktuellen Cursorposition eines RtfEdit-Objekts (obj) ein. Ist eine Selektion vorhanden, wird diese durch das Bild ersetzt. Das Argument Seitennummer (int2) bestimmt bei einem Multipage-TIFF, die Seite, die das einzufügende Bild enthält. Die Seitenzählung beginnt mit 1. Wird (int2) nicht angegeben oder ist der Wert Null, dann wird immer die erste Seite gewählt. Bei Formaten außer TIFF wird das Argument ignoriert.

Optional können in (int3) folgende Optionen angegeben werden:

WinRtfPicModeSpeed Performantes Einfügen

WinRtfPicModeQuality Qualitatives Einfügen

WinRtfPicModeAuto Modus abhängig von der Farbtiefe und Größe des Bildes
auswählen

Wird keine der Optionen angegeben, wird automatisch WinRtfPicModeAuto verwendet.

Die anzeigenbaren Formate sind GIF, TIFF, JPEG, PNG und BMP.



Transparente oder semitransparente Pixel bei PNG bzw. 32bpp BMP-Dateien werden zu weißer Farbe gemischt.

Wird das Bildformat nicht unterstützt, gibt die Funktion den Fehlerwert ErrGeneric zurück. Ist der Arbeitsspeicher nicht ausreichend, wird ErrOutOfMemory zurückgegeben.

Beispiel:

```
// Bild ladentMem # MemAllocate(_MemAutoSize);tFsi # FsiOpen(_Sys->spPathMyPictures + '\MyPicture
```

Mögliche Laufzeitfehler:

Kontakt

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein RtfEdit-Objekt

Kontakt

Befehle für HTTP-Objekte

Liste der Befehle und Konstanten zur Bearbeitung von HTTP-Objekten

Befehlsgruppen,

Siehe Befehlsliste,

HTTP

Befehle

- HttpClose
- HttpGetData
- HttpOpen

Konstanten

- HttpCloseConnection
- HttpDiscard
- HttpParamsToData
- HttpParamsToURI
- HttpProxyAuthTypeBasic
- HttpProxyAuthTypeNTLM
- HttpRecvRequest
- HttpRecvResponse
- HttpSendRequest
- HttpSendResponse
- HttpSkipChunked
- HttpUseWebProxy
- HttpUseWebProxyTLS

HttpOpen(int1, handle2) : 
handle HTTP-Objekt
erzeugen

int1	Objekt-Typ
	<u>HttpSendRequest</u> Anfrage senden
	<u>HttpRecvRequest</u> Anfrage empfangen
handle2	<u>HttpSendResponse</u> Antwort senden
	<u>HttpRecvResponse</u> Antwort empfangen
Resultat handle	Deskriptor einer Socket-Verbindung
	<u>HTTP-Objekt oder</u>  Fehlerwert

Siehe Verwandte Befehle, HTTP,
HttpClose(), Beispiel

Der Befehle erzeugt ein neues HTTP-Objekt vom Typ (int1). In (handle2) muss der Deskriptor der aktiven Socket-Verbindung (SckConnect()) angegeben werden. Das Resultat ist entweder ein Fehlerwert oder der Deskriptor des neuen HTTP-Objekts. Für den Objekttyp gibt es vier Varianten, für die auch unterschiedliche Aktionen durchgeführt werden:

- **HttpSendRequest** - Objekt zum Erstellen und Versenden einer Anfrage

Das Objekt wird mit den Standardwerten für Method, URI und Protocol definiert. Die Listen für Header-Einträge und Parameter sind leer. Das Resultat ist ErrOk.
- **HttpSendResponse** - Objekt zum Erstellen und Versenden einer Antwort

Das Objekt wird mit den Standardwerten für StatusCode und Protocol definiert. Die Listen für Header-Einträge und Parameter sind leer. Das Resultat ist ErrOk.
- **HttpRecvRequest** - Objekt zum Empfangen und Auswerten einer Anfrage

Das Objekt wird angelegt und der komplette Request-Header über den Socket (handle2) in die Liste der Header-Einträge eingelesen. Falls die URI Parameter enthält, werden diese in die Parameterliste übertragen. Das Resultat ist ErrOk, wenn ein gültiger HTTP-Header eingelesen werden konnte. Falls kein gültiger Header ermittelt werden konnte, ist das Resultat ErrData. Als Resultat können auch Socket-Fehler zurückgeliefert werden, wenn beim Lesen der Header-Daten ein solcher Fehler auftritt.
- **HttpRecvResponse** - Objekt zum Empfangen und Auswerten einer Antwort

Das Objekt wird angelegt und der komplette Response-Header über den Socket (handle2) in die Liste der Header-Einträge eingelesen. Das Resultat ist ErrOk, wenn ein gültiger HTTP-Header eingelesen werden konnte. Falls kein gültiger

Kontakt

Header ermittelt werden konnte, ist das Resultat ErrData. Als Resultat können auch Socket-Fehler zurückgeliefert werden, wenn beim Lesen der Header-Daten ein solcher Fehler auftritt.

Mögliche Laufzeitfehler:

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

ErrHdlInvalid Der in (handle2) übergebene Deskriptor ist ungültig.

Kontakt

obj -> HttpGetData(handle1) :



int

Daten des HTTP-Objekts lesen

Deskriptor des

obj

HTTP-Objekts

handle1 Deskriptor des

Ziel-Objekts

Resultat int Fehlerwert



Siehe

Verwandte
Befehle, HTTP,

HttpClose(),

Beispiel

Mit diesem Befehl werden die Daten des HTTP-Bodys eingelesen. Soll der Inhalt des HTTP-Bodys gesetzt werden, erfolgt das mit der Anweisung HttpClose(). In (obj) wird der Deskriptor des HTTP-Objekts übergeben. Das Objekt muss vom Typ HttpRecvRequest oder HttpRecvResponse sein. In (handle1) wird das Objekt angegeben, in das die Daten eingelesen werden sollen. Dafür können drei verschiedene Objekttypen verwendet werden:

- Datei

Der in (handle1) übergeben Deskriptor muss vom Typ HdIFile sein. Die Daten werden direkt in die angegebene Datei geschrieben.

- Memory-Objekt

Der in (handle1) übergeben Deskriptor muss vom Typ HdIMem sein. Die Daten werden in das angegebene Memory-Objekt geschrieben.

- Cte-Liste

Der in (handle1) übergeben Deskriptor muss vom Typ HdICteList sein. In diesem Fall enthält der HTTP-Body Parameter, die in die angegebene Cte-Liste eingelesen werden.



Die Cte-Liste enthält nur dann Werte, wenn die Daten URL-kodiert (x-www-form-urlencoded) übermittelt werden. Ansonsten ist das Ergebnis ErrData.

Neben den Objekttypen müssen noch beim Einlesen von Daten drei Fälle unterschieden werden:

- Im einfachsten Fall hat der HTTP-Header einen Content-Length-Eintrag, der die Größe des HTTP-Bodys enthält. Dadurch ist die Datenmenge bereits vor dem Aufruf der Anweisung bekannt.
- Im zweiten Fall hat der HTTP-Header einen Transfer-Encoding: chunked-Eintrag, der die Übertragung der Daten in mehreren Blöcken anzeigt. Es werden dann soviele Blöcke gelesen, wie die Gegenstelle liefert. In einigen Fällen wird zwischen den Daten und der nächsten Blocklänge kein Zeilenumbruch gesendet. Dies wird ignoriert und der nächste Block gelesen.
- Im dritten Fall werden solange Daten vom Socket gelesen, bis ein Timeout eintritt. Hierbei sollte der Timeout des Sockets vor dem Lesen der Daten auf

Kontakt

einen niedrigen Wert (0,25 bis 3 Sekunden) gesetzt werden, da `HttpGetData()` auf jeden Fall bis zum Timeout wartet.

Als Rückgabewert wird ein Fehlerwert zurückgegeben. Bei `ErrOk` ist kein Fehler aufgetreten. Im Fehlerfall können Fehlerkonstanten aus den Bereichen der externen Dateien, Socketfehler oder Fehler bei der Verarbeitung von `Memory`-Objekten zurückgegeben werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in `(obj)` oder in `(handle1)` übergebene Deskriptor ist ungültig

ErrStringOverflow Das in `(handle1)` angegebene `Memory`-Objekt ist zu klein für die Datenmenge.

Kontakt

obj -> HttpClose(int1[
handle2]) : int



HTTP-Objekt schließen

obj Deskriptor des HTTP-Objekts

Optionen:

HttpDiscard

Daten nicht

versenden

Nach

versenden

Verbindung

trennen

Parameter in

URI schreiben

Parameter in

Body schreiben

Chunked-Daten

überlesen

Senden über

einen

HTTP-Proxy

HttpUseWebProxyTLS Senden über

einen

HTTPS-Proxy

handle2 Deskriptor auf HTTP-Body



Resultat int Fehlerwert

Verwandte Befehle, Http,

Siehe

HttpOpen(), Beispiel

Dieser Befehl schließt das Objekt (obj) und leert die Header- und Parameter-Liste. Alle Objekte, die in den Listen enthalten sind, werden gelöscht.

Bei den Typen HttpRecvRequest und HttpRecvResponse werden eventuell noch ausstehende Daten des HTTP-Body eingelesen und verworfen. Dazu muss der Header über einen Content-Length-Eintrag verfügen.

Bei den Typen HttpSendRequest und HttpSendResponse wird die Anfrage bzw. die Antwort über den bei HttpOpen() angegebenen Socket versendet. Als HTTP-Body kann der Inhalt einer externen Datei, der Inhalt eines Memory-Objekts oder die Parameter-Liste gesendet werden. Der Deskriptor der externen Datei oder des Memory-Objekts wird in (handle2) übergeben. In (int1) können eine oder mehrere der folgenden Optionen angegeben werden:

- HttpDiscard

Die Daten werden nicht versendet.

- HttpCloseConnection

Im HTTP-Header wird der Eintrag Connection: close vorgenommen, der bei HTTP/1.1 dem Empfänger signalisiert, dass die Verbindung nach dem Erhalt der HTTP-Response geschlossen werden soll.

Kontakt

- HttpParamsToURI

Der Inhalt der Parameter-Liste wird in die URI eingefügt. Diese Option ist nur beim Typ HttpSendRequest möglich.

- HttpParamsToData

Der Inhalt der Parameter-Liste wird als HTTP-Body aufbereitet. In (handle2) darf kein Deskriptor angegeben werden.

- HttpSkipChunked

Auf dem Socket vorhandene und noch nicht gelesene Chunked-Daten werden überlesen. Die Option wird nur bei HttpRecvRequest und HttpRecvResponse ausgewertet.

- HttpUseWebProxy / HttpUseWebProxyTLS

Falls das Senden über einen HTTP-Proxy (HttpUseWebProxy) bzw. HTTPS-Proxy (HttpUseWebProxyTLS) erfolgt, muss eine dieser Option verwendet werden. Dadurch wird ein modifizierter HTTP-Header verwendet. Die Eigenschaft ProxyAuthorization ist ebenfalls nur bei diesen Optionen wirksam.

Der Rückgabewert ist der ErrOk, wenn der Versand erfolgreich war. Zu den möglichen Fehlerwerten gehören Socket-Fehler und ErrFsiReadFault, wenn die Datei in (handle2) nicht vollständig gelesen werden kann.

Bei HttpSendRequest werden die folgenden Header-Einträge automatisch gesetzt, wenn sie nicht bereits per Prozedur definiert wurden:

- Host: (falls HostName definiert ist)
- Date:
- User-Agent:

Bei HttpSendResponse werden die folgenden Header-Einträge automatisch gesetzt, wenn sie nicht bereits per Prozedur definiert wurden:

- Date:
- Server:

Der Header-Eintrag Content-Length: wird automatisch gesetzt, wenn ein HTTP-Body vorhanden ist. Der angegebene Deskriptor muss, sofern er nicht weiter benötigt wird, entfernt werden (siehe MemFree() bzw. FsiClose()).

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder in (handle2) übergebene Deskriptor ist ungültig.

ErrStringOverflow Bei der Option HttpParamsToURI wird die URI länger als 8192 Bytes.

Kontakt

Befehle für Job-Verarbeitung

Liste der Befehle und Konstanten zur Verarbeitung von Jobs Siehe
Befehlsgruppen,
Befehlsliste, Job

Befehle

- JobClose
- JobControl
- JobEvent
- JobOpen
- JobSleep
- JobStart

Kontakt

obj -> JobClose()

—————
—————

Kontrollobjekt entfernen
Deskriptor eines
obj

Kontroll-Objekts

Verwandte

Siehe **Befehle**,

JobOpen()

Die Anweisung kann innerhalb einer **Ereignisfunktion des SOA-Service**, sowie im **Standard-** oder **Advanced-Client** ausgeführt werden.

Die Funktion schließt das **JobControl**-Objekt mit dem Deskriptor (obj). Der Job kann in der selben Prozedur oder in einer anderen Ereignisprozedur des Tasks-Prozesses erneut mit **JobOpen()** geöffnet werden, sofern der Job noch aktiv ist oder die beim Starten des Jobs angegebene Zeitspanne nach Jobende noch nicht verstrichen ist.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebene Deskriptor ist ungültig.

Kontakt

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



Job kontrollieren

obj	Deskriptor des JobControl-Objekts
Durchzuführende Funktion	
<u>JobWakeUp</u>	Suspendierten
<u>JobStop</u>	Job aktivieren Job beenden, Ende nicht abwarten
<u>JobTerminate</u>	Job beenden, Ende abwarten
int1	<u>JobMsxTimeoutRead</u> Timeout abfragen
<u>JobEventReceiver</u>	/ setzen Frame-Deskriptor für Job-Events setzen bzw. entfernen
int2	neuer Wert des Timeouts / Frame-Deskriptor

Resultat [int](#) **Ergebnis Text) der Abfrage (siehe**

Siehe [Verwandte Befehle, JobOpen\(\)](#),
[Inter-Thread-Kommunikation](#)
[\(Blog\)](#)

Die Anweisung kann innerhalb einer Ereignisfunktion des SOA-Service, sowie im Standard-Client oder Advanced-Client ausgeführt werden.

Mit dieser Funktion kann ein JobControl- oder Job-Objekt verschiedene Funktionen durchführen. In (obj) wird der Deskriptor des Objektes (siehe JobOpen()) übergeben. In (int1) steht der Typ der durchzuführenden Funktion.

Für JobControl-Objekte sind folgende Optionen verwendbar:

- [JobWakeUp](#)

Mit dieser Option wird der Job aktiviert, wenn er sich in der eigenen Ereignisfunktion mit der Anweisung JobSleep() suspendiert hat. Der Rückgabewert ist ErrTerminated, wenn der Job bereits beendet ist. Ansonsten ist das Resultat ErrOk.

- [JobStop](#)

Diese Funktion setzt die Eigenschaft StopRequest für den Job ohne auf das Ende des Jobs zu warten. Der Rückgabewert ist immer ErrOk.

- [JobTerminate](#)

Diese Funktion setzt die Eigenschaft StopRequest für den Job und wartet darauf, dass sich der Job beendet. Der Rückgabewert ist immer ErrOk.

Kontakt

- JobMsxTimeoutRead

Mit dieser Option kann der Timeout für MsxRead() auf die Message-Pipeline abgefragt (zwei Argumente) oder gesetzt werden (drei Argumente, der neue Wert steht in (int2)). Das Resultat ist der aktuelle bzw. neue Wert des Timeouts in Millisekunden (siehe Verarbeitungshinweise zum SOA-Service).

Für Job-Objekte sind folgende Optionen verwendbar:

- JobMsxTimeoutRead

Mit dieser Option kann der Timeout für MsxRead() auf die Message-Pipeline abgefragt (zwei Argumente) oder gesetzt werden (drei Argumente, der neue Wert steht in (int2)). Das Resultat ist der aktuelle bzw. neue Wert des Timeouts in Millisekunden (siehe Verarbeitungshinweise zum SOA-Service).

- JobEventReceiver

Mit dieser Option kann ein Frame-Deskriptor gesetzt werden (drei Argumente, der Deskriptor steht in (int2)), der durch JobEvent() ausgelöste EvtJob-Ereignisse empfängt. Bei Übergabe von nur zwei Argumenten wird ein zuvor gesetzter Deskriptor wieder entfernt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Deskriptor ist ungültig.

ErrValueInvalid Der in (int1) oder (int2) übergebene Wert ist nicht gültig.

Kontakt

obj -> 

JobEvent([int1])

Job-Ereignis auslösen
Job-Objekt oder

obj

JobControl-Objekt

Minimale
Zeitspanne seit

int1

letztem Aufruf in
Millisekunden

Resultat int Fehlerwert
Verwandte Befehle,

Siehe

EvtJob

Mit dieser Anweisung wird ein Ereignis in die Ereigniswarteschlange des Betriebssystems eingefügt. Das Ereignis wird im Client aufgerufen, wenn die Ereignisse, die vor diesem eingefügt wurden, abgearbeitet sind. Zu welchem Zeitpunkt das Ereignis ausgelöst wird, kann nicht genau bestimmt werden.

Es sind zwei verschiedene Szenarien möglich:

1. Der Aufruf von **JobEvent()** erfolgt in einem **Job**, das Ereignis wird dann in dem Dialog ausgelöst, welcher auf Seite des Jobstarters bei **JobOpen()** angegeben wurde. In (obj) wird das **Job-Objekt** des Jobs angegeben.
2. Der Aufruf von **JobEvent()** erfolgt auf der Seite des Jobstarters, das Ereignis wird dann in dem Dialog ausgelöst, welcher auf Seite des Jobs mittels **JobControl()** und der Option **JobEventReceiver** angegeben wurde. In (obj) wird das **JobControl-Objekt** des Jobstarters angegeben.

Im Parameter (int1) kann optional die minimale Zeitspanne in Millisekunden angegeben werden, die seit dem letzten Aufruf von **JobEvent()** vergangen sein muss. Ist diese noch nicht vergangen, wird der Fehlerwert **ErrLocked** zurückgegeben.

Das Resultat **ErrUnavailable** entsteht, wenn es keinen Empfänger-Dialog für das Ereignis gibt. Dies ist der Fall, wenn bei **JobOpen()** beziehungsweise bei **JobControl()** - mit der Option **JobEventReceiver** - kein Dialog angegeben wurde. Sofern bereits ein **JobEvent** an den Dialog gesendet und noch nicht verarbeitet wurde, gibt die Funktion das Ergebnis **ErrInProgress** zurück.

Beispiel:

```
// Start des Tasks im ClienttJobID # JobStart(_JobThread, 10, 'JobTest:Work');if (tJobID > 0) {
```

Mögliche Laufzeitfehler:

ErrHdIInvalid In (obj) wurde kein gültiger Deskriptor eines **Job-** oder **JobControl-Objektes** übergeben.

Kontakt

JobOpen(int1[,



handle2]) : handle

Kontrollobjekt erzeugen

int1 Id eines Job-Objekts

handle2 Deskriptor eines Dialogs

Deskriptor des

Resultat handle JobControl-Objekts

oder Fehlerwert

Verwandte Befehle,
JobControl, JobClose(),

Siehe

Inter-Thread-Kommunikation

(Blog)

Die Anweisung kann innerhalb einer Ereignisfunktion des SOA-Service, sowie im Standard- oder Advanced-Client ausgeführt werden.

Mit diesem Befehl wird für den Job mit der Id (int1) ein Kontrollobjekt angelegt. Pro Job kann nur ein Kontrollobjekt vorhanden sein. Das Resultat ist ErrInUse, wenn bereits ein Kontrollobjekt für diesen Job besteht. Existiert kein Job mit der angegebenen Id, wird ErrUnknown zurückgeliefert. Das Kontrollobjekt sollte vor Ende der Prozedur mit JobClose() wieder geschlossen werden.

Im Parameter (handle2) kann optional der Deskriptor eines Dialoges angegeben werden. Ereignisse, die mit JobEvent() von dem Job ausgeführt werden, werden an das Ereignis EvtJob von dem angegebenen Dialog weitergeleitet.



Der Parameter (handle2) kann nur beim Aufruf aus dem Standard- oder Advanced-Client angegeben werden.

Kontakt

JobSleep(int1) :



logic

Job suspendieren

Wartezeit in
int1

Millisekunden

Resultat logic Wartezustand abgebrochen

Siehe [Verwandte Befehle](#),

[JobControl\(\)](#)

Die Anweisung kann innerhalb eines [Jobs](#) ausgeführt werden. Jobs können durch eine [Ereignisfunktion des SOA-Service](#), einen [Standard-](#) oder [Advanced-Client](#) gestartet werden.

Diese Funktion hält die Verarbeitung im Job für (int1) Millisekunden an. Im Unterschied zu [SysSleep\(\)](#) kann der Wartezustand abgebrochen werden. In diesem Fall wird als Resultat [true](#) zurückgegeben. Ist die Wartezeit ohne Unterbrechung verstrichen, so ist das Resultat [false](#). Ein Abbruch findet statt, wenn der komplette Task gestoppt werden soll oder wenn der Job durch [JobControl\(..., JobWakeUp\)](#) aufgeweckt wird.

Mögliche Laufzeitfehler:

[ErrValueInvalid](#) In (int1) wurde ein Wert < 0 angegeben.

Kontakt

JobStart(int1, int2,
alpha3[, alpha4[,
alpha5[, handle6]]]) : 

int

Job starten

Ausführungstyp:

JobThread Thread innerhalb des

Prozesses

JobProcess Betriebssystemprozess

JobCopyBuffers Inhalt der globalen
Feldpuffer kopieren (nur

int1 vom Client aus)

JobWinPrt Verwendung von
Oberflächenobjektbefehlen
ermöglichen (nur vom
Client aus)

JobStartNoRTE Laufzeitfehler bei JobStart

unterbinden

Zeitspanne, in der die

int2 Job-Informationen zur Verfügung
stehen

alpha3 Name der Ereignisfunktion

alpha4 Argumente für den Job (optional)

alpha5 Beschreibung des Jobs (optional)

handle6 Socket-Deskriptor (optional)

Job-Id oder Fehlerwert

Wert > 0 Job-Id des
neuen Jobs

Resultat int ErrTerminated Job
erfolgreich
durchgeführt

anderer Wert < Fehlerwert

0

Siehe Verwandte Befehle, JobOpen()

Die Anweisung kann innerhalb einer Ereignisfunktion des SOA-Service, sowie im Standard-
oder Advanced-Client ausgeführt werden.

Mit diesem Befehl wird ein neuer Job gestartet. In (int1) wird der Ausführungstyp des Jobs
angegeben: JobThread startet einen neuen Job-Thread innerhalb des aktuellen
Betriebssystemprozesses, JobProcess einen neuen Betriebssystemprozess für diesen Job (siehe
Verarbeitungshinweise).



Der Ausführungstyp JobProcess steht nur im SOA-Service zur Verfügung. Wird sie im
Standard- oder Advanced-Client angegeben, wird der Laufzeitfehler ErrValueInvalid
ausgelöst.

Der Ausführungstyp JobThread kann im Standard- oder Advanced-Client mit der Optionen
JobCopyBuffers und JobWinPrt kombiniert werden.

Kontakt

Ist die Option JobCopyBuffers angegeben, werden beim Start des Jobs die aktuellen Feldpuffer des Clients in die Feldpuffer des Jobs übertragen.



Wurde zwischen Anmelden des Benutzers an der Datenbank und dem JobStart(_JobThread | _JobCopyBuffers, ...) die Datenstruktur neu aufgebaut, wird der Job nicht gestartet und die Eigenschaft JobErrorCode auf den Fehler ErrIllegalOp gesetzt.

Die Option JobWinPrt ermöglicht die Verwendung von Dialogen und Oberflächenobjektbefehlen im Job.



Ein mit JobStart() gestarteter Thread ist von anderen Threads isoliert. Somit kann nur auf Eigenschaften und Ereignisse von Oberflächenobjekten des eigenen Threads zugegriffen werden. Zur Thread-Kommunikation können die Message-Exchange-Befehle verwendet werden. Zum Signalisieren von Nachrichten kann der Befehl JobEvent() verwendet werden.



Die Option JobWinPrt sollte nur verwendet werden, wenn auch auf Oberflächenobjektbefehle zugegriffen wird.

In (int2) wird die Zeitspanne in Sekunden angegeben, in der der Job noch nach seiner Beendigung mit JobOpen() geöffnet werden kann. Bei einem Wert von 0 kann der Job geöffnet werden, solange er ausgeführt wird. Der Maximalwert für (int2) beträgt 86400 Sekunden (24 Stunden).

In (alpha3) befindet sich der Name der Ereignisfunktion, die der Job aufrufen soll. In (alpha4) kann optional ein Argumenttext übergegeben werden, der in der Jobprozedur über die Eigenschaft JobData abgerufen werden kann. In (alpha5) kann optional ein Beschreibungstext für den Job angegeben werden, der in der Jobprozedur über die Eigenschaft SvcDescription abgerufen werden kann. Die Argumente und die Beschreibung können jeweils bis zu 8192 Zeichen lang sein.

Optional kann in (handle6) ein Socket-Deskriptor (siehe SckConnect()) angegeben werden. Dabei wird eine Kopie des Deskriptors erstellt, der auf die gleiche Socket-Verbindung verweist. Diese ist in der Job-Funktion über die Eigenschaft JobSckHandle zugreifbar. Die Socket-Verbindung kann dann sowohl in dem Job, als auch in der aktuellen Prozedur verwendet werden. Somit kann beispielsweise im Job auf neue Nachrichten gewartet und gleichzeitig in der aktuellen Prozedur Nachrichten versendet werden. Hierbei sollte jedoch eine Seite nur lesen und eine nur schreiben.

Die Socket-Verbindung muss nur auf der Seite geschlossen werden, die JobStart() aufgerufen hat. Wird der Socket auf der Seite des Jobs geschlossen, wird die Verbindung auf beiden Seiten getrennt. Jedoch bleibt der Socket-Deskriptor in der Prozedur, die den Job gestartet hat, erhalten. Dieser muss dann separat mit SckClose() geschlossen werden. Am Ende des Jobs wird die erzeugte Kopie des Socket-Dekriptors entfernt. Die Socket-Verbindung bleibt am Ende des Jobs erhalten und kann in der aufrufenden Prozedur bis zum SckClose() verwendet werden.



Es kann kein Socket-Deskriptor angegeben werden, der mit einer der _SckTls...- Optionen erzeugt wurde.

Der Rückgabewert ist die Job-Id des neuen Jobs (Wert > 0). Ist der Job bereits durchgeführt, bevor JobStart() zurück gekommen ist, oder soll der Client, der Soa-Service oder der aktuelle Job beendet werden, wird ErrTerminated

Kontakt

zurückgegeben. Falls beim Start des Jobs ein Fehler aufgetreten ist, enthält der Rückgabewert den Fehlerwert (Wert < 0). Die Job-Id ist innerhalb des laufenden Tasks eindeutig. Sie wird zum Erzeugen eines Kontroll-Objekts mit JobOpen() benötigt.



Ist die Option (int1) JobStartNoRTE angegeben, werden keine Laufzeitfehler ausgelöst.
Statt dessen wird der Wert des Laufzeitfehlers als Fehlerwert zurückgeliefert.

Mögliche Laufzeitfehler:

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

ErrNoProcInfo Prozedur ist nicht vorhanden oder wurde nicht übersetzt

ErrNoSub Prozedurfunktion ist nicht vorhanden

Kontakt

Befehle für PDF-Verarbeitung

Liste der Befehle und Konstanten zur Verarbeitung von PDF-Dateien

[Befehlsgruppen](#),

Siehe [Befehlsliste](#),

[PDF](#)

Befehle

- [PdfAttachExportFile](#)
- [PdfAttachExportMem](#)
- [PdfAttachFile](#)
- [PdfAttachInfoGet](#)
- [PdfAttachMem](#)
- [PdfClose](#)
- [PdfInsertImage](#)
- [PdfInsertMeta](#)
- [PdfNew](#)
- [PdfOpen](#)
- [PdfPageClose](#)
- [PdfPageOpen](#)
- [PdfSavePage](#)
- [PdfTextColor](#)
- [PdfTextExtractMem](#)
- [PdfTextFont](#)
- [PdfTextWrite](#)

Kontakt

obj -> PdfAttachExportFile(int1, alpha2) : int



Anhang eines PDF-Dokumentes in eine Datei exportieren

 Deskriptor des

obj

 PDF-Objekts

 Nummer des

int1

 Anhangs

alpha2 Name der zu

 Zieldatei

—

Resultat int Fehlerwert

Verwandte Befehle,

PdfAttachInfoGet(),

Siehe PdfAttachExportMem(),

PdfAttachCount,

PdfAttachFile()

Der Befehl exportiert eine Anhangdatei eines PDF-Dokumentes (obj). Das PDF muss zuvor mit der Anweisung PdfOpen() geöffnet worden sein.

Im Argument (int1) muss die Nummer der Anhangdatei angegeben werden. Diese Nummer muss zwischen 1 und Der Anzahl Anhänge (siehe PdfAttachCount) liegen.

Die Zieldatei wird in (alpha2) angegeben. Diese Datei wird erzeugt bzw.
überschrieben.

Fehlerwerte

Zusätzlich zu den Fehlerwerten für externe Dateioperationen (ErrFsi...) kann von der Funktion ErrGeneric zurückgegeben werden, wenn es einen Fehler beim Auslesen der Dateiinformationen gab.

Beispiel:

```
// Anhänge zähltentAttachCount # tHdlPdf->spPdfAttachCount;// Anhänge durchgehenfor tAttachNo #
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Deskriptor ist ungültig.

ErrValueInvalid Ungültige Anhangsnummer (int1) angegeben.

Kontakt



obj -> PdfAttachExportMem(int1, handle2) : int
Anhang eines PDF-Dokumentes in eine Datei exportieren
obj Deskriptor des
PDF-Objekts
int1 Nummer des
Anhangs
handle2 Deskriptor eines
Memory-Objektes
Resultat int Fehlerwert 
Verwandte Befehle,
PdfAttachInfoGet(),
Siehe PdfAttachExportFile(),
PdfAttachCount,
PdfAttachMem()

Der Befehl liest eine Anhangdatei eines PDF-Dokumentes (obj) in ein Memory-Objekt ein. Das PDF muss zuvor mit der Anweisung PdfOpen() geöffnet worden sein.

Im Argument (int1) muss die Nummer der Anhangdatei angegeben werden. Diese Nummer muss zwischen 1 und Der Anzahl Anhänge (siehe PdfAttachCount) liegen.

Mit dem Argument (handle2) wird das Memory-Objekt angegeben, in welches die Anhangdatei geschrieben wird. Das Memory-Objekt muss groß genug sein oder mit MemAutoSize angelegt werden. Der Inhalt des Memory-Objektes wird überschrieben.

Fehlerwerte

Von der Funktion kann ErrGeneric zurückgegeben werden, wenn es einen Fehler beim Auslesen der Dateiinformationen gab.

Beispiel:

```
// Anhänge zähltentAttachCount # tHdlPdf->spPdfAttachCount;// Anhänge durchgehenfor tAttachNo #
```

Mögliche Laufzeitfehler:

- ErrHdlInvalid Der in (obj) bzw. (handle2) angegebene Deskriptor ist ungültig.
- ErrValueInvalid Ungültige Anhangsnummer (int1) angegeben.
- ErrMemExhausted Das Memory-Objekt (handle2) konnte nicht vergrößert werden.

Kontakt

ErrOk kein Fehler aufgetreten

ErrFsiNoFile Pfad- oder Dateiname nicht gefunden

ErrOutOfMemory Arbeitsspeicher nicht ausreichend

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Deskriptor ist ungültig.

ErrValueInvalid Kein Dateiname (alpha1) angegeben oder kein zulässiger Wert für die Verknüpfungsart (int2).

Kontakt



obj -> PdfAttachInfoGet(int1, int2) : alpha

Informationen über einen Anhang eines PDF-Dokumentes ermitteln

obj Deskriptor des PDF-Objekts

int1 Nummer des Anhangs

Informationstyp
PdfAttachInfoName Name des

Anhangs

PdfAttachInfoDescription Beschreibung

des Anhangs
Größe des

int2 PdfAttachInfoSize Anhangs in Byte

Erstellungsdatum

des Anhangs

PdfAttachInfoTimeCreate Erstellungszeit

des Anhangs

PdfAttachInfoDateModify Änderungsdatum

des Anhangs

PdfAttachInfoTimeModify Änderungszeit

des Anhangs

Resultat alpha Information 

Siehe Verwandte Befehle, PdfAttachExportFile(),

PdfAttachExportMem(), PdfAttachCount

Der Befehl liest Informationen einer Anhangdatei eines PDF-Dokumentes (obj) aus.

Das PDF muss zuvor mit der Anweisung PdfOpen() geöffnet worden sein.

Im Argument (int1) muss die Nummer der Anhangdatei angegeben werden. Diese Nummer muss zwischen 1 und Der Anzahl Anhänge (siehe PdfAttachCount) liegen.

Zusätzlich wird in (int2) der Informationstyp angegeben. Folgende Werte können angegeben werden:

PdfAttachInfoName Name des Anhangs

PdfAttachInfoDescription Beschreibung des Anhangs

PdfAttachInfoSize Größe des Anhangs in Byte

PdfAttachInfoDateCreate Erstellungsdatum des Anhangs im internen Format (siehe FmtInternal)

PdfAttachInfoTimeCreate Erstellungszeit des Anhangs im internen Format (siehe FmtInternal)

PdfAttachInfoDateModify Änderungsdatum des Anhangs im internen Format (siehe FmtInternal)

PdfAttachInfoTimeModify Änderungszeit des Anhangs im internen Format (siehe FmtInternal)

Der Rückgabewert enthält die ermittelten Informationen des Anhangs. Im Fehlerfall wird der globale Fehlerwert gesetzt. Gab es einen Fehler beim Auslesen der Dateiinformationen, wird der globale Fehlerwert auf ErrGeneric gesetzt.

Beispiel:

Kontakt

```
// Anhänge zähltentAttachCount # tHdlPdf->spPdfAttachCount;// Anhänge durchgehenfor      tAttachNo #
```

Mögliche Laufzeitfehler:

- ErrHdlInvalid** Der in (obj) angegebene Deskriptor ist ungültig.
 Ungültige Anhangsnummer (int1) oder ungültiger Informationstyp
ErrValueInvalid (int2) angegeben.

Kontakt

obj -> PdfAttachMem(handle1,
alpha2[, int3[, alpha4[, logic5]]]) |  : int
Anhangdatei zu PDF hinzufügen
Deskriptor des
obj PDF-Objekts
handle1 Memory-Objekt der
Anhangdatei
Name der
alpha2 Anhangdatei (nur
intern)
Verknüpfungsart der
int3 Anhangdatei
Beschreibung der
alpha4 Anhangdatei
Anhang
komprimieren
Resultat int Fehlerwert 
Verwandte Befehle,
PdfAttachFile(),
Siehe PdfAttachExportMem(),
Blog

Der Befehl fügt einem PDF eine extern vorliegende Anhangdatei hinzu. Das PDF muss zuvor mit der Anweisung PdfOpen() geöffnet worden sein.

handle1 ist der Deskriptor des Memory-Objektes mit den Anhangdaten.

Im Argument (alpha2) wird ein interner Name für die einzufügende Anhangdatei angegeben.

Mit dem Argument (int3) kann die Verknüpfungsart definiert werden. Folgende Arten können angegeben werden:

- PdfAttachAssociateAlternative

Verknüpfung des Anhangs mit Beziehung "Alternative". Die Konstante sollte ab sofort anstelle von PdfAttachAssociateZUGFeRD verwendet werden.

- PdfAttachAssociateSource

Verknüpfung des Anhangs mit Beziehung "Source".

Mit dem Argument (alpha4) kann eine Beschreibung für den Anhang angegeben werden.

Mit dem Argument (logic5) wird definiert, ob die Anhangdatei komprimiert werden soll.



Eine Komprimierung führt nicht immer zu einer Verkleinerung des PDF. Im Extremfall ist das resultierende PDF mit Komprimierung größer als ohne. Dies hängt von den im Anhang enthaltenen Daten ab. Textdateien lassen sich meist

Kontakt

gut komprimieren, wohingegen bereits komprimierte Dateien (z. B. ZIP-Dateien) sich nicht weiter komprimieren lassen.

Der Rückgabewert kann mit folgenden Konstanten verglichen werden:

ErrOk kein Fehler aufgetreten

ErrOutOfMemory Arbeitsspeicher nicht ausreichend

Mögliche Laufzeitfehler:

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

ErrValueInvalid Kein Dateiname (alpha2) angegeben oder kein zulässiger Wert für die Verknüpfungsart (int3).

Kontakt

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



int

PDF-Objekt schließen

obj Deskriptor des PDF-Objekts

alpha1 Dateiname der zu schreibenden Datei (optional)

Schreibmodus (optional)

PdfModePdfNormal

normales PDF

PdfModePdfA

erzeugen
PDF/A-1b:2005

PdfModePdfANormal

erzeugen
Versuchen
PDF/A-1b:2005 zu

PdfModePdfA2b

erzeugen
PDF/A-2b:2011

PdfModePdfA2bNormal

erzeugen
Versuchen
PDF/A-2b:2011 zu

PdfModePdfA3b

erzeugen
PDF/A-3b:2012

PdfModePdfA3bNormal

erzeugen
Versuchen
PDF/A-3b:2012 zu

PdfModeCancel

erzeugen
keine Datei erzeugen
ZUGFeRD-konformes
PDF (Basic)

PdfModePdfZUGFeRDBasic

int2

PdfModePdfZUGFeRDComfort

erzeugen
ZUGFeRD-konformes
PDF (Comfort)

PdfModePdfZUGFeRDExtended

erzeugen
ZUGFeRD-konformes
PDF (Extended)

PdfModePdfZUGFeRD10

erzeugen
ZUGFeRD
1.0-konformes PDF

PdfModePdfZUGFeRD20

erzeugen
ZUGFeRD
2.0-konformes PDF

PdfModePdfZUGFeRD21

erzeugen
ZUGFeRD
2.1-konformes PDF

PdfModePdfZUGFeRXRechnung ZUGFeRD

erzeugen
2.1-konformes PDF
mit
XRechnung-Profil

Resultat int

Fehlerwert

Kontakt

Siehe [Verwandte Befehle](#), [Blog](#)

Mit dieser Anweisung wird ein zuvor mit [PdfNew\(\)](#) erstelltes bzw. [PdfOpen\(\)](#) geöffnetes PDF-Dokument zurückgeschrieben und das Objekt entfernt. Alle an dem Dokument durchgeführten Änderungen werden erst zu diesem Zeitpunkt in die Datei geschrieben.

Der Deskriptor des PDF-Dokuments wird als (obj) angegeben. Die zu schreibende Datei kann in (alpha1) übergeben werden. Wird kein Dateiname angegeben, wird das Dokument nicht geschrieben. In (int2) kann eine der folgenden Konstanten angegeben werden:

- [PdfModePdfNormal](#)

Es wird ein normales PDF-Dokument erzeugt.

- [PdfModePdfA](#)

Es wird ein PDF/A-1b:2005 konformes Dokument erzeugt.

- [PdfModePdfANormal](#)

Es wird versucht ein PDF/A-1b:2005 konformes Dokument zu erzeugen. Gelingt dies nicht, wird ein normales PDF-Dokument erstellt und [ErrPdfNotPdfA](#) zurückgegeben.

- [PdfModePdfA2b](#)

Es wird ein PDF/A-2b:2011 konformes Dokument erzeugt.

- [PdfModePdfA2bNormal](#)

Es wird versucht ein PDF/A-2b:2011 konformes Dokument zu erzeugen. Gelingt dies nicht, wird ein normales PDF-Dokument erstellt und [ErrPdfNotPdfA](#) zurückgegeben.

- [PdfModePdfA3b](#)

Es wird ein PDF/A-3b:2012 konformes Dokument erzeugt.

- [PdfModePdfA3bNormal](#)

Es wird versucht ein PDF/A-3b:2012 konformes Dokument zu erzeugen. Gelingt dies nicht, wird ein normales PDF-Dokument erstellt und [ErrPdfNotPdfA](#) zurückgegeben.

- [PdfModePdfZUGFeRDBasic](#)

Erstellung eines ZUGFeRD-konformen PDF mit Basic-Profil.

- [PdfModePdfZUGFeRDComfort](#)

Erstellung eines ZUGFeRD-konformen PDF mit Comfort-Profil.

- [PdfModePdfZUGFeRDExtended](#)

Erstellung eines ZUGFeRD-konformen PDF mit Extended-Profil.

- [PdfModePdfZUGFeRD10](#)

Erstellung eines ZUGFeRD Version 1.0-konformen PDF.

- [PdfModePdfZUGFeRD20](#)

Kontakt

Erstellung eines ZUGFeRD Version 2.0-konformen PDF.

- PdfModePdfZUGFeRD21

Erstellung eines ZUGFeRD Version 2.1-konformen PDF.

- PdfModePdfZUGFeRDXRechnung

Erstellung eines ZUGFeRD Version 2.1-konformen PDF mit XRechnung-Profil.

- PdfModeCancel

Die Bearbeitung des PDF-Dokuments wird beendet, ohne das Dokument zu speichern.

Für ZUGFeRD-konforme PDFs kann je eine der Profil-Konstanten

(PdfModePdfZUGFeRDBasic, PdfModePdfZUGFeRDComfort,
PdfModePdfZUGFeRDExtended) mit je einer der Versions-Konstanten
(PdfModePdfZUGFeRD10, PdfModePdfZUGFeRD20, PdfModePdfZUGFeRD21)
kombiniert werden. Ist eine ZUGFeRD Profil-Konstante, jedoch keine
Versions-Konstante angegeben, wird die Version 1.0 verwendet.



Die Konformität des XML-Anhangs beim Generieren einer ZUGFeRD-PDF-Datei wird nicht gewährleistet. Die XML-Datei kann mit Schema-Dateien von FeRD überprüft werden. Fertige PDF-Dateien können hier validiert werden.

Folgende Fehlercodes können zurückgegeben werden:

<u>ErrOk</u>	Kein Fehler aufgetreten.
<u>ErrGeneric</u>	Es ist ein nicht spezifizierte Fehler aufgetreten.
<u>ErrPdfNotPdfA</u>	Es konnte kein PDF/A-konformes Dokument erstellt werden.
<u>ErrOutOfMemory</u>	Der zur Erstellung des PDF-Dokuments notwendige Hauptspeicher konnte nicht allokiert werden.
<u>ErrFsiOpenFailed</u>	Beim erstellen der externen Datei ist ein Fehler aufgetreten.
Mögliche Laufzeitfehler:	
<u>ErrHdlInvalid</u>	Der in (obj) angegebene Deskriptor ist ungültig.
<u>ErrValueInvalid</u>	In (int2) wurde ein ungültiger Wert angegeben.

Kontakt

obj -> PdfInsertImage(alpha1, float2, float3, float4,

float5, int6[, int7]) : int

Bild einfügen (BMP, GIF, JPEG, PNG und TIFF)

Deskriptor des

obj PDF-Objekts

Pfad- und

alpha1 Dateiname des

Bildes

Abstand vom

float2 linken Seitenrand

Abstand vom

float3 oberen Seitenrand

float4 Breite des Bildes

float5 Höhe des Bildes

int6 Seitennummer

Größenanpassung
des Bildes oder

int7 des PDFs

(optional)

Resultat int Fehlerwert

Verwandte

Siehe

Befehle

Die Anweisung fügt ein Bild der Formate BMP, GIF, JPEG, PNG oder TIFF in die zum Bearbeiten geöffnete Seite ein. Die Seite muss zuvor mit der Anweisung PdfPageOpen() geöffnet werden. Das Bild wird an der angegebenen Position (float2 und float3) in der angegebenen Größe (float4 und float5) eingefügt. Die Positions- und Größenangaben erfolgen in Millimeter.

Beinhaltet die Bild-Datei mehr als ein Bild (zum Beispiel bei einem Multipage-TIFF), wird der Index des Bildes in (int6) übergeben.

Soll eine Anpassung der Größe des Bildes oder der Seite des PDF-Dokuments vorgenommen werden, kann eine der folgenden Konstanten in (int7) angegeben werden:

- _PdfInsertNormal

Das Bild wird ohne Anpassungen des Bildes oder des PDF-Dokuments eingefügt.

- _PdfInsertFitImage

Das Bild wird an das Pdf angepasst. Es wird entweder an die Seitenhöhe oder die Seitenbreite angepasst, sodass das komplette Bild in der maximalen Größe dargestellt wird. Das Verhältnis der Höhe zur Breite bleibt bestehen.

- _PdfInsertFitPdf

Das Pdf wird auf die Größe des Bildes angepasst. Dabei kann bei den Parametern (float4) und (float5) angegeben werden, ob die Originalgröße des Bildes beibehalten werden soll (-1.0), oder es können die gewünschten Werte in Millimeter übergeben werden.

- _PdfOptUseLogSize



Kontakt

Wird diese Konstante mit der Konstanten `_PdfInsertFitImage` oder `_PdfInsertFitPdf` kombiniert, wird beim Skalieren die logische Größe des Bildes berücksichtigt. Bei Kombination mit `_PdfInsertFitPdf` muss die Breite (`float4`) und die Höhe (`float5`) jeweils -1.0 sein.

-  Die eingefügten Bilder werden anhand der übergebenen Argumente zwischengespeichert. Bei erneutem Aufruf mit den selben Argumenten, wird das bereits gespeicherte Bild eingefügt. Die Größe und das Änderungsdatum der Datei werden hierbei nicht berücksichtigt.

Der Rückgabewert kann mit folgenden Konstanten verglichen werden:

<u>ErrOk</u>	kein Fehler aufgetreten
<u>ErrFsiNoFile</u>	Pfad- oder Dateiname nicht gefunden
<u>ErrOutOfMemory</u>	Arbeitsspeicher nicht ausreichend
<u>ErrPdfPageClosed</u>	keine Seite geöffnet
<u>ErrPdfImageFormat</u> nicht unterstütztes Bildformat	
<u>ErrGeneric</u>	anderer Fehler aufgetreten

Mögliche Laufzeitfehler:

<u>ErrHdlInvalid</u>	Der in (obj) angegebene Deskriptor ist ungültig.
--------------------------------------	--

Kontakt

**obj -> PdfInsertMeta(alpha1,
float2, float3, float4, float5) :**
int

Bild einfügen (WMF und EMF)

Deskriptor
obj des
PDF-Objekts
Pfad- und
alpha1 Dateiname
des Bildes
Abstand vom
float2 linken
Seitenrand
Abstand vom
float3 oberen
Seitenrand
Breite des
float4 Bildes
Höhe des
float5 Bildes

Resultat int Fehlerwert
Verwandte
Siehe
Befehle

Die Anweisung fügt ein Windows Metafile (WMF) oder ein Enhanced-Metafile (EMF) in die zum Bearbeiten geöffnete Seite ein. Die Seite muss zuvor mit der Anweisung [PdfPageOpen\(\)](#) geöffnet werden. Das Bild wird an der angegebenen Position (float2 und float3) in der angegebenen Größe (float4 und float5) eingefügt. Die Positions- und Größenangaben erfolgen in Millimeter.

Konnte das Bild eingefügt werden, wird [ErrOk](#), sonst [ErrPdfInsertMetaFile](#) zurückgegeben.

Mögliche Laufzeitfehler:

[ErrHdlInvalid](#) Der in (obj) angegebene Deskriptor ist ungültig.



Kontakt

PdfNew([alpha1[, alpha2]]) : 

handle PDF-Dokument

erstellen

alpha1 Kennwort des Eigentümers

alpha2 Kennwort zum Öffnen

Resultat handle

PDF-Objekts

Verwandte Befehle, PDF,

Siehe PdfClose(), PdfOpen(),

PDF-Verarbeitung (Blog)

Diese Anweisung öffnet ein neues PDF-Objekt. Der Deskriptor auf das Objekt wird von der Anweisung zurückgegeben. Das PDF-Dokument kann mit einem Eigentümer-Kennwort und/oder einem Öffnen-Kennwort versehen werden, diese können in (alpha1) und (alpha2) angegeben werden. Das Dokument wird erst mit der Anweisung PdfClose() extern gespeichert.

Wurde ein Eigentümer- oder Öffnen-Kennwort angegeben, können die Berechtigungen über die Eigenschaft PdfUserRights gesetzt werden. Ohne die Angabe eines Kennwortes wird diese Eigenschaft nicht ausgewertet.

Standardmäßig werden Seiten in dem Dokument in der Größe DinA4 angelegt. Die Seitengröße kann über die Eigenschaften PdfPageWidth und PdfPageHeight definiert werden.

PdfOpen(alpha1[,
int2, alpha3[, int4]]) : 

handle

PDF-Datei öffnen

alpha1 Pfad- und Dateiname

Art des Kennworts (optional)

_PdfOpenDefault Das

Open-Kennwort

int2

ist angegeben

_PdfOpenOwner Das

Owner-Kennwort

ist angegeben

alpha3 Kennwort (optional)

Inhalte des PDF-Dokuments

_PdfImportDefault Import ohne
dynamische

int4

Inhalte

_PdfImportAll Import aller

Inhalte

Deskriptor des

Resultat **handle PDF-Objekts**

Verwandte Befehle, PDF,

Siehe PdfClose(), PdfNew(),

PDF-Verarbeitung (Blog)

Diese Anweisung öffnet die in (alpha1) angegebene PDF-Datei für die Verarbeitung. Kann die Datei geöffnet werden, wird der Deskriptor auf das PDF-Objekt zurückgegeben.

Ist das PDF-Dokument durch ein Kennwort geschützt, kann das Kennwort in (alpha3) angegeben werden. (int2) bestimmt, ob das Kennwort zum Öffnen des Dokuments (_PdfOpenDefault) oder das Kennwort des Eigentümers (_PdfOpenOwner) übergeben wurde.

In (int4) kann der zu importierende Inhalt aus dem PDF-Dokument eingeschränkt werden:

- _PdfImportDefault

Es werden keine dynamischen Inhalte, wie zum Beispiel Formularfelder, importiert.

- _PdfImportAll

Es werden alle Inhalte des PDF-Dokuments importiert. Das kann dazu führen, dass JavaScript-Aktionen durchgeführt werden, wenn das Dokument JavaScript enthält.

Tritt beim Öffnen des Dokuments ein Fehler auf, wird einer der folgenden Fehlerwerte zurückgegeben:

Kontakt

<u>ErrFsiNoFile</u>	Der Pfad oder die Datei ist nicht vorhanden.
<u>ErrFsiInvalidFormat</u>	Die Datei ist nicht im PDF-Format oder ist defekt.
<u>ErrPdfPassword</u>	Es wurde das falsche Owner- bzw. Open-Kennwort bei einer verschlüsselten PDF-Datei angegeben.
<u>ErrOutOfMemory</u>	Es steht nicht genügend Hauptspeicher zum Öffnen der Datei zur Verfügung.
<u>ErrGeneric</u>	Es ist ein anderer Fehler aufgetreten.

Mögliche Laufzeitfehler:

<u>ErrValueInvalid</u>	Der in (int2) angegebene Parameter ist ungültig.
-------------------------------	--

Kontakt

obj -> PdfPageClose([int1]) 

Bearbeitete Seite schließen

 Deskriptor
 obj des
 PDF-Objekts
 Seite
 speichern
 int1 oder
 verwerfen
 (optional)

Siehe **Verwandte Befehle**

Mit der Anweisung wird eine Seite, die zuvor mit [PdfPageOpen\(\)](#) zur Bearbeitung geöffnet wurde, wieder geschlossen. Als (obj) wird der Deskriptor des entsprechenden PDF-Objekts übergeben.

In (int1) kann angegeben werden, ob die Seite gespeichert (_PdfPageCloseNormal) oder die Änderungen verworfen (_PdfPageCloseCancel) werden sollen. Wird kein Parameter angegeben, wird die Seite gespeichert. In jedem Fall wird die Seite aus dem Speicher entfernt und steht nicht mehr für die entsprechenden Anweisungen für die Bearbeitung von PDF-Seiten zur Verfügung.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

Kontakt

obj -> PdfPageOpen(int1) : int  Seite
anlegen oder bearbeiten

 Deskriptor
obj eines
 PDF-Objekts
 Nummer der
int1 Seite

Resultat int Fehlerwert

Siehe Verwandte

Befehle

Mit dieser Anweisung wird eine Seite eines zuvor mit PdfOpen() geöffneten oder PdfNew() angelegten PDF-Dokuments zur Bearbeitung geöffnet. Die Seitennummer wird in (int1) angegeben. Ist die Seite noch nicht vorhanden, wird sie erzeugt.

Von einem Dokument kann immer nur eine Seite zur Bearbeitung geöffnet werden.

Konnte die Seite geöffnet werden wird ErrOk, sonst ErrGeneric zurückgegeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Desktiptor ist ungültig.

Kontakt

PdfSavePage(alpha1, alpha2, int3[, point4[, point5[, int6[, alpha7]]]])  : int

Grafikdatei aus einer Seite des PDF-Dokuments erzeugen
Dateiname der

alpha1 PDF-Datei

alpha2 Dateiname der
Grafik-Datei

int3 Seitennummer

point4 Größe des Bildes
(optional)

Auflösung des

point5 Bildes (optional)
Qualität des

int6 Bildes (nur
JPEG) (optional)

Kennwort der

alpha7 PDF-Datei
(optional)

Resultat int Fehlerwert  —

Verwandte

Siehe

Befehle

Der Befehl erzeugt eine Grafik-Datei aus einer Seite einer PDF-Datei. Der Pfad und der Name der PDF-Datei wird in (alpha1) angegeben. Die Angabe der zu erzeugenden Grafik-Datei erfolgt in (alpha2). Abhängig von der angegebenen Dateinamen-Erweiterung wird eine Datei des entsprechenden Formats erzeugt.

- .jpg, .jpeg - JPEG-Grafik
- .tif, .tiff - TIFF-Grafik
- .png - PNG-Grafik
- .bmp - Windows-Bitmap

Bei anderen Erweiterungen oder wenn keine Erweiterung vorhanden ist, wird als Resultat ErrGeneric zurückgegeben.



Ist die in (alpha2) angegebene Datei bereits vorhanden, wird sie überschrieben.

Die Seite, die in ein Bild gewandelt werden soll, wird in (int3) angegeben. Die Nummer muss im Bereich 1 bis Anzahl der Seiten im PDF liegen. Bei einem Wert außerhalb liefert die Anweisung das Resultat ErrPdfPageNotExisting.

Die Größe des Bildes wird in (point4) angegeben. Wird das Argument nicht angegeben oder (0, 0) übergeben, wird die Seitengröße der PDF-Seite verwendet.

In (point5) kann die Auflösung des resultierendes Bildes angegeben werden. Die Komponente :x enthält dabei die horizontale, die Komponente :y die vertikale Auflösung. Wird das Argument nicht angegeben oder (0, 0) übergeben, wird ein Bild mit 72x72 DPI generiert. Ist nur einer der beiden Werte 0, dann ist die Auflösung in beiden Dimensionen identisch. D. h. bei der Übergaben von (300, 0) wird ein Bild mit der Auflösung 300x300 DPI erzeugt.

Kontakt

Handelt es sich bei der erzeugten Grafik um eine JPEG-Datei, kann in (int6) die Qualität des Bildes angegeben werden. Der Wert muss im Bereich 1 (schlecht) bis 100 (gut) liegen. Liegt er außerhalb wird er entsprechend modifiziert. Die Angabe wird bei anderen Grafik-Formaten ignoriert.

Ist das in (alpha1) angegebene PDF-Dokument durch ein Kennwort geschützt, muss dieses Kennwort in (alpha7) angegeben werden.

Die Anweisung gibt einen Fehlerwert zurück. Der Wert kann mit folgenden Konstanten verglichen werden:

<u>ErrOk</u>	Datei wurde erfolgreich erstellt.
<u>ErrOutOfMemory</u>	Speicher nicht ausreichend.
<u>ErrFsiInvalidFormat</u>	Keine PDF-Datei oder PDF-Datei korrupt.
<u>ErrPdfPassword</u>	PDF-Datei ist kennwortgeschützt bzw. Kennwort falsch.
<u>ErrGeneric</u>	Ein unbekannter Fehler ist aufgetreten. Dieser Fehler wird auch zurückgegeben, wenn die Dateierweiterung der Ausgabedatei nicht zulässig ist.
<u>ErrPdfPageNotExisting</u>	Die angegebene Seite existiert nicht.
<u>ErrFsiOpenFailed</u>	Die Ausgabedatei konnte nicht angelegt werden.
<u>ErrFsiWriteFault</u>	Fehler beim Schreiben der Ausgabedatei.

Kontakt

obj ->
PdfTextColor(int1) : int
Farbe für Text festlegen
 Deskriptor
obj des
 PDF-Objekts
 Farbe des
int1
 Textes



Resultat int Fehlerwert
Verwandte

Siehe
Befehle

Diese Anweisung setzt die Farbe des Fonts, der für die nachfolgenden PdfTextWrite()-Anweisungen verwendet wird. Als (obj) wird ein PDF-Objekt angegeben. Die Farbe wird als RGB-Wert oder mit einer der Farbkonstanten (WinColBlack, WinColBlue, WinColGreen, WinColRed, WinColYellow, WinColMagenta, WinColCyan, WinColWhite, WinColDarkGray, WinColLightBlue, WinColLightGreen, WinColLightRed, WinColLightYellow, WinColLightMagenta, WinColLightCyan, WinColLightGray) angegeben.

 Die Farbe muss nach dem Öffnen einer Seite im PDF-Dokument angegeben werden und ist bis zum Schließen der Seite gültig. Bei der Bearbeitung mehrerer Seiten, muss nach jedem Öffnen einer Seite mit der Anweisung PdfPageOpen() die Farbe gesetzt werden.>

Konnte die Farbe gesetzt werden, gibt die Anweisung ErrOk, sonst ErrGeneric zurück.

Beispiel:

```
...tHdlPdf # PdfOpen(_Sys->spPathMyDocuments + '\Document.pdf');tHdlPdf->PdfPageOpen(1);tColor->v
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

Kontakt

obj -> PdfTextExtractMem(handle1) : int 

Textinformationen einer PDF-Seite ermitteln

obj
Deskriptor eines

PDF-Objekts

handle1 Deskriptor eines
Memory-Objekts

Resultat int Fehlerwert

Verwandte

Siehe Befehle,

PdfPageOpen()

Mit dieser Anweisung wird die Textinformation einer zuvor mit PdfPageOpen() geöffneten Seite eines PDF-Dokuments ermittelt und in das Memory-Objekt (handle1) übertragen.

Ist keine Seite geöffnet, gibt die Anweisung den Fehlerwert ErrPdfPageClosed zurück. Der ermittelte Text wird an das Memory-Objekt angehängt (siehe Len). Der Zeichensatz des Memory-Objekts wird entsprechend der Eigenschaft Charset berücksichtigt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Desktiptor ist ungültig.

Kontakt

**obj -> PdfTextFont(alpha1,
float2[, int3[, logic4]]) : int**



Font für Text festlegen

obj Deskriptor des PDF-Objekts

alpha1 Name des Fonts

float2 Größe des Font

Font-Attribute (optional)

_WinFontAttrNormal Normale

Darstellung

int3 **_WinFontAttrItalic**

Kursiv

_WinFontAttrBold

Fett

_WinFontAttrUnderline Unterstrichen

_WinFontAttrStrikeOut Durchgestrichen

logic4 Font einbetten (optional)

Resultat int Fehlerwert

Siehe **Verwandte Befehle**

Diese Anweisung lädt einen Font, der für die nachfolgenden **PdfTextWrite()**-Anweisungen verwendet wird. Als (obj) wird ein PDF-Objekt angegeben. Der Name des Fonts wird in (alpha1), seine Größe in (float2) angegeben. Die Angabe der Größe erfolgt in Millimetern.



Der Font muss nach dem Öffnen einer Seite im PDF-Dokument angegeben werden und ist bis zum Schließen der Seite gültig. Bei der Bearbeitung mehrerer Seiten, muss nach jedem Öffnen einer Seite mit der Anweisung **PdfPageOpen()** der Font gesetzt werden.

In (int3) können die Font-Attribute angegeben werden. Sind keine Font-Attribute angegeben, erfolgt die normale Darstellung. Es kann eine Kombination aus folgenden Konstanten angegeben werden:

_WinFontAttrNormal Normale Darstellung

_WinFontAttrItalic Kursiv

_WinFontAttrBold Fett

_WinFontAttrUnderline Unterstrichen

_WinFontAttrStrikeOut Durchgestrichen

Über den Parameter (logic4) kann bestimmt werden, ob der Font in das Dokument eingebunden wird (**true**), oder nicht (**false**). Standardmäßig wird der Font nicht eingebunden. Eingebundene Fonts müssen auf dem System, auf dem das Dokument angezeigt werden soll, nicht vorhanden sein. Ist ein Font nicht eingebunden und nicht auf dem System installiert, wird ein ähnlicher Font zur Anzeige des Textes durch das System gewählt.

Konnte der Font gesetzt werden, gibt die Anweisung **ErrOk**, sonst **ErrGeneric** zurück.

Beispiel:

```
tHdlPdf # PdfOpen(_Sys->spPathMyDocuments + '\Document.pdf');tHdlPdf->PdfTextFont('Arial', 5.0, _
```

Kontakt

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

Kontakt

obj ->
PdfTextWrite(float1, float2, alpha3)
float1, float2, alpha3
: int
Text schreiben
 Deskriptor
obj des
 PDF-Objekts
 Abstand vom
float1 linken
 Seitenrand
 Abstand vom
float2 oberen
 Seitenrand
 zu
alpha3 schreibender
 Text
Resultat int Fehlerwert
 Siehe
 Verwandte
 Befehle

Diese Anweisung schreibt einen Text in ein PDF-Dokument. In (obj) wird der Deskriptor des PDF-Objekts angegeben. Die zu bearbeitende Seite muss zuvor mit [PdfPageOpen\(\)](#) geöffnet worden sein. Die Position des Textes wird in den Parametern (float1) und (float2) in Millimetern angegeben. Der zu schreibende Text wird in (alpha3) übergeben.

Der zu verwendende Font und die Textfarbe müssen zuvor mit den Anweisungen [PdfTextFont\(\)](#) und [PdfTextColor\(\)](#) definiert werden.

Konnte der Text geschrieben werden, gibt die Anweisung [ErrOk](#), sonst [ErrGeneric](#) zurück.

Beispiel:

```
tHd1Pdf # PdfOpen(_Sys->spPathMyDocuments + '\Document.pdf');if (tHd1Pdf->PdfPageOpen(1) = _ErrOk
```

Mögliche Laufzeitfehler:

[ErrHd1Invalid](#) Der in (obj) übergebene Deskriptor ist ungültig.

Kontakt

Befehle für Diagramme

Liste der Befehle zur Bearbeitung von Diagrammen

Siehe Befehlsgruppen Befehlsliste,

Befehle

- ChartClose
- ChartDataAdd
- ChartDataClose
- ChartDataOpen
- ChartDataSort
- ChartOpen
- ChartSave

Erstellen von Diagrammen

Ein Diagramm wird aus zwei verschiedenen Objekten erstellt. Den äußeren Rahmen eines Diagramms stellt das Chart-Objekt zur Verfügung. In den Eigenschaften dieses Objekts werden der Typ des Diagramms und alle Form gebenden Charakteristika angegeben. Die darzustellenden Daten werden in einem oder mehreren ChartData-Objekten angegeben. Die Darstellung einer Datenreihe erfolgt dabei auf Basis der Eigenschaften des Chart-Objekts. Durch Änderung der Eigenschaften können auch unterschiedliche Darstellungen der Datenreihen erfolgen.

Ein Diagramm wird mit der Anweisung ChartOpen() erzeugt. Dabei wird der Diagrammtyp, die Größe des Ausgabebereiches und der Titel des Diagramms angegeben.

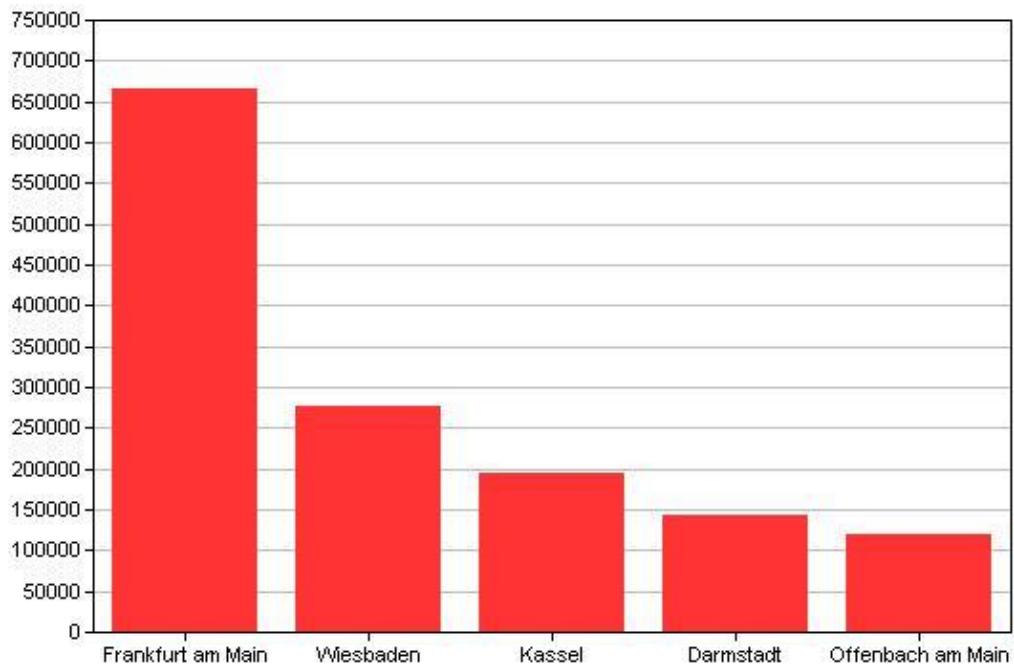
Anschließend wird ein ChartData-Objekt erzeugt. Diesem Objekt werden mit der Anweisung ChartDataAdd() die darzustellenden Daten zugewiesen. In dem folgenden Beispiel sind das die Einwohnerzahlen der fünf größten Städte Hessens (Stand 2008) und die Bezeichner der einzelnen Daten (die Namen der Städte). Die Zuweisung kann ebenfalls über Arrays erfolgen, wird hier aber der Einfachheit wegen einzeln durchgeführt.

Die Ausgabe des Diagramms erfolgt in eine externe Datei (ChartSave()).

Die Möglichkeiten bei der Erstellung von Diagrammen sind in einem Beispiel in der Datenbank "CodeLibrary" ausgeführt. Die Beschreibung der Eigenschaften befinden sich im Abschnitt Eigenschaften eines Chart-Objekts.

```
@A+@C+mainlocal{ tChart      : handle;  tChartData : handle;}{  tChart # ChartOpen(_ChartXY, 800,
```

Die fünf größten Städte Hessens



Kontakt

**ChartOpen(int1, int2, int3[,
alpha4[,int5]]) : handle**
Chart-Grafik anlegen



Typ der Chart-Grafik		
	<u>_ChartPie</u>	Torten-Diagramm
int1	<u>_ChartXY</u>	Koordinaten-Diagramm
	<u>_ChartPyramid</u>	Pyramiden-Diagramm
	<u>_ChartSurface</u>	Oberflächen-Diagramm
int2	Breite des <u>Chart-Objekts</u>	
int3	Höhe des <u>Chart-Objekts</u>	
alpha4	Titel der Grafik (optional)	
	Legende (optional)	
	<u>_ChartOptDefault</u>	keine
	<u>_ChartOptLegendVertical</u>	Legende vertikal angeordnete
int5		Legende
	<u>_ChartOptLegendHorizontal</u>	horizontal angeordnete Legende

Resultat handle

Chart-Objekts

Siehe Verwandte Befehle, ChartClose()

Mit dieser Anweisung wird ein Chart-Objekt angelegt. Der Typ der Grafik wird im Parameter (int1) angegeben. Folgende Konstanten stehen dafür zur Verfügung:

- _ChartPie

Es wird eine Tortengrafik angelegt. Es können Torten- oder Donut-Grafiken erzeugt werden. Donut-Grafiken werden dabei durch die zusätzliche Angabe eines Innenradius in der Eigenschaft ChartPieInnerRadius erstellt. Die einzelnen Daten werden als Teil der Gesamtmenge angezeigt.

- _ChartXY

Es wird ein Koordinaten-Diagramm angezeigt. Es können Balken-, Linien- oder Flächen-Grafiken erzeugt werden. Die Darstellung wird über die Eigenschaft ChartXYStyleData bestimmt.

- _ChartPyramid

Es wird ein Pyramiden-Diagramm angezeigt. Es können Pyramiden-, Kegel- oder Trichter-Grafiken erzeugt werden. Die Darstellung wird über die Eigenschaft ChartPyramidStyleData bestimmt.

- _ChartSurface

Es wird ein Oberflächen-Diagramm angezeigt.

Die Parameter (int2) und (int3) bestimmen die Breite und Höhe der Grafik. Die Angaben erfolgen in Pixel. In (alpha4) kann ein Titel für die Grafik angegeben werden. Der Titel wird oben innerhalb der Grafik angezeigt.

Kontakt

Standardmäßig wird keine Legende angezeigt. Soll eine Legende mit gezeichnet werden, kann zwischen folgenden Typen ausgewählt werden:

- _ChartOptDefault
Es wird keine Legende gezeichnet.
- _ChartOptLegendVertical
Es wird eine vertikale Legende gezeichnet.
- _ChartOptLegendHorizontal
Es wird eine horizontale Legende gezeichnet.

Die Position, Darstellung und Inhalt der Legende werden über verschiedene Eigenschaften des Chart-Objekts gesteuert.

Die Grafik wird erst dann erzeugt, wenn das Chart-Objekt mit der Anweisung ChartSave() gespeichert wurde. Die Speicherung kann entweder in eine externe Datei oder in einem Memory-Objekt erfolgen. Anschließend kann das Chart-Objekt mit der Anweisung ChartClose() geschlossen werden.

Die darzustellenden Daten werden mit einem ChartData-Objekt angegeben. Das Objekt kann mit der Anweisung ChartDataOpen() erzeugt werden.

Bei erfolgreicher Durchführung gibt die Anweisung den Deskriptor auf ein Chart-Objekt zurück. Steht zum Anlegen des Objekts nicht genug Speicher zur Verfügung, wird die Konstante ErrOutOfMemory zurückgegeben.

Beispiel:

```
// PietHdlChart # ChartOpen(_ChartPie, 400, 300, 'Title');...tHdlChart->ChartSave(_Sys->spPathTem
```

Mögliche Laufzeitfehler:

ErrValueInvalid Im Parameter (int1) wurde ein ungültiger Typ oder in (int5) ein ungültiger Wert angegeben.

Kontakt

obj -> ChartSave(alpha1[, int2[, handle3]]) : int Speichern des



Chart-Objekts

obj Deskriptor auf Chart-Objekt

alpha1 Pfad- und Dateiname

Format des zu erzeugenden Bildes

(optional)

_ChartFormatAuto Format aus

Dateierweiterung

_ChartFormatPng Speicherung als

PNG

int2 _ChartFormatJpg Speicherung als

JPEG

_ChartFormatGif Speicherung als

GIF

_ChartFormatBmp Speicherung als

Bitmap

handle3 Deskriptor eines Memory-Objekts

—

Resultat int Fehlerwert

Siehe Verwandte Befehle, ChartOpen()

Mit diesem Befehl wird ein Chart-Objekt als Bild in einer externen Datei oder einem Memory-Objekt gespeichert. Im (obj) wird der Deskriptor übergeben, der von ChartOpen() zurückgegeben wurde. In (alpha1) wird der Pfad und der Name der externen Datei übergeben. Die Angabe einer externen Datei kann entfallen, wenn die Speicherung in einem Memory-Objekt erfolgt.

In (int2) wird das Dateiformat angegeben. Wird dieser Parameter nicht angegeben, wird das Dateiformat aus der Dateierweiterung des externen Dateinamens ermittelt (entspricht _ChartFormatAuto). Folgende Konstanten können angegeben werden:

- _ChartFormatAuto

Das Dateiformat wird in Abhängigkeit von der Dateierweiterung des in (alpha1) angegebenen Dateinamens gewählt. Zulässige Dateierweiterungen sind .png, .jpg, .jpeg, .gif und .bmp.

- _ChartFormatPng

Es wird eine PNG-Datei erzeugt.

- _ChartFormatJpg

Es wird eine JPEG-Datei erzeugt.

- _ChartFormatGif

Es wird eine GIF-Datei erzeugt.

- _ChartFormatBmp

Es wird eine Bitmap-Datei erzeugt.

- _ChartFormatPdf

Kontakt

Es wird eine PDF-Datei erzeugt.

Erfolgt die Speicherung in einem Memory-Objekt, muss in (handle3) der entsprechende Deskriptor angegeben werden. Das Memory-Objekt muss ausreichend dimensioniert sein, dass die Ausgabedaten komplett hineinpassen. Hat das Objekt die Option MemAutoSize gesetzt, wird es automatisch vergrößert, wenn notwendig. Die Daten werden an die aktuelle Position im Memory-Objekt angehängt. Ist als Format ChartFormatAuto angegeben, werden Daten im PNG-Format erzeugt. Ist in (handle3) ein Wert angegeben, wird der Dateiname in (alpha1) ignoriert.

 Nach der Durchführung der Anweisung haben Änderungen an Eigenschaften und Chart-Daten keine Auswirkung mehr. Es ist jedoch möglich den Befehl mehr als einmal aufzurufen, um mehrere Dateien unterschiedlicher Formate zu generieren.

Wurden die Daten erfolgreich gespeichert, gibt die Anweisung ErrOk zurück. Ist beim Schreiben in eine externe Datei ein Fehler aufgetreten, wird ErrFsiOther zurückgegeben.

Mögliche Laufzeitfehler:

ErrValueInvalid In (alpha1) wurde eine leere Zeichenkette oder in (int2) ein ungültiger Wert angegeben. Der Laufzeitfehler wird auch generiert, wenn ChartFormatAuto angegeben ist, der Dateiname jedoch keine gültige Endung besitzt.

ErrHdlInvalid Bei dem in (obj) übergebenen Deskriptor handelt es sich nicht um einen gültigen Chart-Deskriptor oder bei (handle3) nicht um ein Memory-Objekt.

ErrValueRange Das übergebene Memory-Objekt hat nicht genug Platz zum Schreiben aller Daten.

Kontakt

obj -> ChartClose()

Chart-Grafik schließen
Deskriptor des
obj

Chart-Objekts

Verwandte

Siehe **Befehle**,

ChartOpen()

Mit diesem Befehl wird ein **Chart**-Objekt geschlossen, wenn es nicht mehr benötigt wird. Im Parameter (obj) wird der durch **ChartOpen()** zurückgegebene Deskriptor angegeben.

Der Speicher, des Chart-Objekts und der dazugehörenden Daten wird freigegeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebene Deskriptor ist kein gültiges **Chart**-Objekt.

Kontakt

obj -> ChartDataOpen(int1[, int2]) : handle Objekt



für Daten einer Chart-Grafik anlegen

obj Deskriptor eines Chart- oder
ChartData-Objekts

int1 Maximale Anzahl an Werten

Informationstypen (optional)

_ChartDataValue Datenmenge

_ChartDataLabel Beschriftungen

int2 _ChartDataColor Füllfarben für

Datenbereiche

_ChartDataExtra Zusätzliche

alphanumerische Daten

Resultat handle Deskriptor des ChartData-Objekts

—
Verwandte Befehle, ChartDataClose(),

Siehe

ChartDataAdd()

Die Anweisung öffnet ein neues ChartData-Objekt. In diesem Objekt werden Daten zur Anzeige in einem Chart-Objekt angegeben. Dabei können neben den darzustellenden Daten, Bezeichnungen und Farben angegeben werden.

Im Argument (obj) wird der Deskriptor des Chart-Objekts übergeben. Dieser wird von der Anweisung ChartOpen() zurück gegeben. In (int1) wird angegeben, wieviel Werte das Chart-Objekt maximal enthalten soll. Die später tatsächlich dargestellte Anzahl kann darunter liegen, darf diese Obergrenze jedoch nicht überschreiten.

Das Argument (int2) bestimmt die Art der Daten, die das ChartData-Objekt transportieren kann. Jedes Chart benötigt eine Datenmenge. Diese repräsentiert die darzustellenden Daten. Die Konstante hierfür lautet _ChartDataValue. Da die Datenmenge obligatorisch ist, muss sie nicht angegeben werden.

Neben der Datenmenge gibt es noch weitere Datentypen. Folgende Konstanten können als Datentypen angegeben werden:

- _ChartDataValue

Mit dieser Konstante wird das Objekt zum Transport der Datenmenge vorbereitet. Da jedes Chart-Objekt einen Datenmenge zur Anzeige benötigt, muss diese Konstante beim Erzeugen des ChartData-Objekts nicht angegeben werden. Beim Hinzufügen von Daten ist das der Default-Wert.

- _ChartDataLabel

Mit dieser Konstante wird das Objekt zum Transport von Beschriftungen für die einzelnen Daten vorbereitet. Diese Beschriftungen werden neben den Sektoren (Darstellung als Tortendiagramm, ChartPie) oder unter der X-Achse (Darstellung als Koordinaten-Diagramm, ChartXY) angezeigt.

- _ChartDataColor

Mit dieser Konstante wird das Objekt zum Transport von Farben für die einzelnen Daten vorbereitet. Die Sektoren (Darstellung als Torten-Diagramm, ChartPie) bzw. die Balken (Darstellung als Koordinaten-Diagramm, ChartXY)

Kontakt

werden in diesen Farben gezeichnet.

- ChartDataExtra

Mit dieser Konstante können zusätzliche alphanumerische Daten angegeben werden. Beim Koordinaten-Diagramm (ChartXY) dienen sie zur Anzeige einer Beschriftung zum entsprechenden Wert der Datenmenge.

Das Objekt wird mit der Anweisung ChartDataOpen() zur Aufnahme der entsprechenden Daten vorbereitet. Sollen in den späteren ChartDataAdd()-Anweisungen neben den Daten noch Bezeichner, Farben und/oder zusätzliche Werte übertragen werden, muss in der Anweisung eine entsprechende Kombination aus den Konstanten angegeben werden.

Chart-Objekte vom Typ ChartXY können mehr als eine Datenmenge besitzen. Die zusätzlichen Datenmengen stellen dann weitere Reihen von Balken, Linien- oder Flächen dar. Als Bezeichner für die X-Achse wird der Inhalt der letzten Datenreihe verwendet. Damit die zusätzlichen Beschriftungen angezeigt werden, muss in der Eigenschaft ChartXYStyleLabel des Chart-Objektes die Konstante ChartXYStyleLabelDataExtra gesetzt sein. Die Textfarbe lässt sich über ChartXYLabelColData beeinflussen. Die Textrotation geschieht über die Eigenschaft ChartXYLabelRotData.

ChartData-Objekte von den Typen ChartXY und ChartSurface können untergeordnete Datenmengen besitzen. Bei gestapelten (ChartXYStyleDataStack) und prozentualen (ChartXYStyleDataPercent) Koordinatendiagrammen ist jede untergeordnete Datenreihe eine Ebene in die Höhe. Die Farbe der Datenreihe wird von der ersten Farbe in der untergeordneten Datenreihe bestimmt. Bei Streudiagrammen (ChartXYStyleDataScatter) müssen 2 bis 3 untergeordnete Datenreihen angegeben werden. Die erste gibt Koordinaten auf der x-Achse, die zweite auf der y-Achse an. Sind 3 untergeordnete Datenreihen vorhanden, bestimmt die dritte Datenreihe die Größe der Punkte in Pixeln. Ist diese nicht angegeben, werden alle Daten in der Größe der Eigenschaft ChartXYLineSymbolSize dargestellt.

Bei Oberflächen-Diagrammen (ChartSurface) müssen 3 untergeordnete Datenmengen angegeben werden. Diese definieren die Daten der x-, y- und z-Achse. Auf der z-Achse sollten $x * y$ Daten angegeben werden. Um die Farben der Oberfläche zu beeinflussen, müssen diese in der Hauptdatenmenge angegeben werden. Die Angabe von Beschriftungen (ChartDataLabel) ist bei der ersten und zweiten untergeordneten Datenmenge möglich.



Untergeordnete Datenmengen können nicht mit ChartDataSort() sortiert oder mit ChartDataClose() geschlossen werden.

Die eigentlichen Daten, Bezeichner und Farben werden durch die Anweisung ChartDataAdd() eingefügt.

Beispiele:

Es werden nur Daten übermittelt:

```
tHd1Chart # ChartOpen(_ChartPie, 400, 300);tHd1ChartData # tHd1Chart->ChartDataOpen(20);...
```

Kontakt

Es werden Daten, Bezeichner und Farben übermittelt:

```
tHd1Chart # ChartOpen(_ChartPie, 400, 300);tHd1ChartData # tHd1Chart->ChartDataOpen(20, _ChartDat
```

Mehrere Datenreihen:

```
tHd1Chart # ChartOpen(_ChartXY, 400, 300); // first rowtHd1ChartData # tHd1Chart->ChartDataOpen(20
```

Datenreihen für Oberflächen-Diagramme:

```
tHd1Chart # ChartOpen(_ChartSurface, 400, 300);tHd1ChartDataMain # tHd1Chart->ChartDataOpen(10, _
```

Die Funktion liefert bei erfolgreicher Durchführung einen Deskriptor auf ein ChartData-Objekt.

Kann das Objekt wegen Speichermangel nicht angelegt werden, wird die Konstante

ErrOutOfMemory zurückgegeben.

Mögliche Laufzeitfehler:

ErrHd1Invalid Bei dem übergebenen Deskriptor in (obj) handelt es sich nicht um einen gültigen Deskriptor eines Chart-Objekts.

ErrValueInvalid Das Argument (int2) enthält einen ungültigen Wert oder das Argument (int1) ist negativ.

Kontakt

obj -> ChartDataAdd(var1[, int2]) : int Daten



in ein ChartData-Objekt einfügen

obj Deskriptor auf ein
ChartData-Objekt

var1 Einzufügende Daten

Datentyp
ChartDataValue Datenmenge
ChartDataLabel Beschriftungen
ChartDataColor Farben
ChartDataExtra Zusätzliche

Daten

Resultat int Fehlerwert
Siehe Verwandte Befehle,



ChartDataOpen()

Mit dieser Anweisung werden Daten einem ChartData-Objekt hinzugefügt. Das Objekt muss zuvor mit ChartDataOpen() erzeugt worden sein. Der Deskriptor des Objekts wird in (obj) übergeben.

In (var1) werden die Daten übergeben. Abhängig vom Typ der Daten kann hier eine Variable bzw. ein Array mit bestimmten Datentypen angegeben werden. Der Typ der Daten wird in (int2) übergeben:

Typ der Daten (int2)	(var1)	Beschreibung
<u>ChartDataValue</u>	<u>int</u> , <u>bigint</u> , <u>float</u>	darzustellende Daten
<u>ChartDataLabel</u>	<u>alpha</u>	Bezeichner der Sektoren oder Balken
<u>ChartDataColor</u>	<u>color</u> , <u>int</u>	Farben der Sektoren oder Balken
<u>ChartDataExtra</u>	<u>alpha</u>	Zusätzliche Daten

Farbwerte von Datentyp color besitzen neben den Werten für die Anteile Rot, Grün und Blau zusätzlich einen Transparenzwert, mit dem die Deckung der Farbe angegeben werden kann.

Wird (int2) nicht angegeben, werden in (var1) darzustellende Daten angegeben. Die Datenmenge kann in Form von int-, bigint- oder float-Werten übergeben werden. Eine Mischung der Daten in einer Datenreihe ist nicht möglich. Als Bezeichnungen können nur Zeichenketten (alpha) angegeben werden. Farben können als _WinCol...-Farbwerte oder mit den Datentyp color angegeben werden. Auch hier ist eine Mischung der Datentypen nicht möglich.

Wird als Wert die Konstante ChartDataNoValue angegeben, kennzeichnet sie einen fehlenden Wert.

Daten vom Typ ChartDataLabel, ChartDataColor und ChartDataExtra können nur übergeben werden, wenn das ChartData-Objekt mit der Anweisung ChartDataOpen() entsprechend vorbereitet wurde.

Konnte die Anweisung ausgeführt werden, gibt sie den Wert ErrOk zurück. Steht kein ausreichender Speicher zur Verfügung, wird ErrOutOfMemory zurückgegeben.

Kontakt

Beispiel:

```
tChartData->ChartDataAdd('Frankfurt am Main', _ChartDataLabel);tChartData->ChartDataAdd(664838);t
```

Mögliche Laufzeitfehler:

In (int2) wurde ein ungültiger Wert übergeben oder der angegebene Datentyp wurde nicht bei ChartDataOpen() spezifiziert. Der

ErrValueInvalid Laufzeitfehler wird auch generiert, wenn die Obergrenze der insgesamt möglichen Daten (siehe ChartDataOpen()) überschritten wurde.

ErrFldType

In (var1) wurde ein nicht unterstützter Datentyp übergeben.

Kontakt

obj -> ChartDataSort([int1[, range2]]) : int



Daten im ChartData-Objekt sortieren

obj Deskriptor des ChartData-Objekts

Sortier-Optionen (optional)

_ChartDataSortLabel Sortierung nach der

Beschriftung

_ChartDataSortLabelCI Sortierung nach der

Beschriftung (ohne

Unterscheidung der

Groß-/Kleinschreibung)

int1

_ChartDataSortValue Sortierung der

Datenmenge

(aufsteigend)

range2 Bereich (optional)

Resultat int ErrOk



Siehe Verwandte Befehle, ChartDataAdd()

Mit dieser Anweisung werden die Daten in einem ChartData-Objekt sortiert. Der Deskriptor des Daten-Objekts wird in (obj) angegeben. Im Parameter (int1) wird das Sortier-Kriterium angegeben. Folgende Optionen stehen zur Verfügung:

- _ChartDataSortLabel

Die Daten werden nach den Bezeichnern sortiert.

- _ChartDataSortLabelCI

Die Daten werden nach den Bezeichnern sortiert. Die Groß-/Kleinschreibung wird dabei nicht berücksichtigt.

- _ChartDataSortValue

Die Daten werden nach den Werten sortiert. Die Sortierung erfolgt aufsteigend.

Die Zuordnung von Bezeichnern, Farben und Daten bleiben durch die Sortierung der Daten erhalten. Die Optionen können nicht miteinander kombiniert werden.

Im optionalen Parameter (range2) kann ein Bereich angegeben werden, wenn nur eine Teilmenge sortiert werden soll. Ohne Angabe des Arguments findet keine Einschränkung statt.



ChartData-Objekte, die anderen ChartData-Objekten untergeordnet sind, können nicht sortiert werden.

Beispiele:

```
// Nach Beschriftung sortieren ohne Bereichseinschränkung tChartData->ChartDataSort(_ChartDataSor
```

Als Rückgabewert wird immer ErrOk zurückgegeben.

Mögliche Laufzeitfehler:

Kontakt

Der in (int1) angegebene Datentyp ist nicht im ChartData-Objekt

ErrValueInvalid enthalten oder der angegebene Bereich liegt nicht innerhalb der bei
ChartDataOpen() angegebenen Anzahl der Elemente.

ErrHdlInvalid In (obj) wurde kein gültiger ChartData-Deskriptor angegeben.

Kontakt

obj -> ChartDataClose()  Daten-Objekt für
Chart-Grafik schließen
obj Deskriptor eines
ChartData-Objekts

Siehe [Verwandte Befehle](#),
[ChartDataOpen\(\)](#)

Mit dem Befehl wird ein zuvor durch [ChartDataOpen\(\)](#) geöffnetes [ChartData-Objekt](#) geschlossen.

In (obj) wird der Deskriptor auf das [ChartData-Objekt](#) übergeben. Die [ChartData-Objekte](#) eines [Chart-Objekts](#) werden automatisch geschlossen, wenn der Befehl [ChartClose\(\)](#) des [Chart-Objekts](#) aufgerufen wird.

Beim Schließen des [ChartData-Objekts](#) werden Datenmenge und alle weiteren Daten (Farben und Beschriftungen) an das [Chart-Objekt](#) übertragen. Änderungen an den Daten sind anschließend nicht mehr möglich. Danach kann der Befehl [ChartSave\(\)](#) aufgerufen werden, um eine Ausgabe des Charts in einer externen Datei oder einem [Memory-Objekt](#) zu erhalten.

Verschiedene Eigenschaften des [Chart-Objekts](#) wirken sich erst beim Schließen einer Datenreihe aus. Dies ist bei den entsprechenden [Eigenschaften](#) gesondert vermerkt.

 [ChartData-Objekte](#), die anderen [ChartData-Objekten](#) untergeordnet sind, können nicht explizit geschlossen werden. Sie werden automatisch beim Schließen des übergeordneten [ChartData-Objektes](#) geschlossen.

Die Anweisung liefert zur Zeit immer den Wert [ErrOk](#) zurück.

Mögliche Laufzeitfehler:

ErrHdIInvalid In (obj) wurde kein gültiger Deskriptor auf ein [ChartData-Objekt](#) übergeben.

Schnittstellen und Kommunikation
Schnittstellen und Kommunikation
Siehe **Befehlsgruppen**, **Befehlsliste**,

- [Befehle für JSON-Verarbeitung](#)
- [Befehle für ODBC-Verbindungen](#)
- [Befehle für XML-Verarbeitung](#)
- [ClipBoard-Befehle](#)
- [COM-Befehle](#)
- [DDE-Befehle](#)
- [DLL-Befehle](#)
- [E-Mail-Befehle](#)
- [Socket-Befehle](#)
- [TAPI-Befehle](#)
- [Web-Schnittstellen-Befehle](#)

Kontakt

Clipboard-Befehle

Befehle für den Datenaustausch über die Zwischenablage von Windows [Verwandte](#)

[Befehle, Liste](#)

[sortiert nach](#)

Siehe [Gruppen](#),

[Alphabetische](#)

[Liste aller](#)

[Befehle](#)

Befehle

- [ClipboardRead](#)

- [ClipboardWrite](#)

Kontakt

ClipBoardRead() : alpha |  Windows-

Zwischenablage abrufen

Inhalt der

Resultat alpha

Windows-Zwischenablage

Siehe [ClipBoardWrite\(\)](#), [TextClipBoard](#)

Dieser Befehl liefert als Resultat einen alpha-Wert, der den Inhalt der Windows-Zwischenablage enthält.

Kontakt

ClipBoardWrite(alpha1)  Text in die Windows-Zwischenablage übertragen

alpha1 Text
Siehe [ClipBoardRead\(\)](#),
[TextClipboard](#)

Mit dieser Funktion wird der Inhalt von (alpha1) in die Windows-Zwischenablage übertragen.

Kontakt

Befehle für XML-Verarbeitung

Liste der Befehle und Konstanten zur Verarbeitung von XML Siehe

Befehlsgruppen Befehlsliste,

Sobald Informationen aus einer Applikation in einer externen Datei gespeichert werden sollen, stellt sich die Frage nach dem Format der externen Datei. Grundsätzlich kann zwischen zwei Alternativen gewählt werden. Die Informationen können in einem binären Format oder in einem Textformat abgelegt werden. Beide Formen bieten Vor- und Nachteile. Die Speicherung im Textformat ist häufig wesentlich größer als ein binäres Format, kann aber leichter durch den Programmierer überprüft werden.

Zur Speicherung von strukturierten Daten in einem Textformat stellt XML (Extensible Markup Language) eine Reihe von Regeln zur Verfügung, mit deren Hilfe das Format einer Datei definiert werden kann. Es gibt somit nicht das XML-Format, sondern vielmehr eine ganze Reihe von Formaten, die mit Hilfe von XML erzeugt wurden. Mit XML wird dabei festgelegt, wie bestimmte Daten gespeichert werden. Die Interpretation der Daten erfolgt durch die Applikation. Eine Beschreibung der Regeln befinden sich auf der Web-Präsenz des W3-Konsortiums: www.w3.org.

Alle XML-verarbeitenden Programme müssen die Zeichenkodierung UTF-8 und UTF-16 unterstützen. Sollen die Daten mit anderen Applikationen ausgetauscht werden, muss eine dieser Zeichenkodierungen verwendet werden. Es können aber auch andere Kodierungen angegeben werden.

Ein XML-Dokument ist immer hierarchisch aufgebaut. Der Dokument-Knoten bildet dabei das Wurzel-Element. Alle weiteren Elemente sind baumartig dem Wurzel-Element untergeordnet. Jedes Element besteht aus einer Start- und einer Ende-Zeichenkette. Die Eltern-Kind-Beziehung wird dadurch gekennzeichnet, dass sich das Kind-Element vollständig zwischen den Start- und Ende-Zeichenketten des Eltern-Elements befindet.

In CONZEPT 16 wird diese hierarchische Struktur mit Hilfe von CteNode-Objekten abgebildet. Um eine XML-Datei zu erzeugen, wird die hierarchische Struktur mit Cte-Objekten erzeugt und anschließend mit der Anweisung XmlSave() in eine externe Datei geschrieben. Umgekehrt wird beim Lesen einer XML-Datei mit XmlLoad() die Struktur in Cte-Objekte überführt und kann mit CteRead() durchsucht werden.

Die Informationen innerhalb der Datei werden in den Knoten-Objekten abgelegt. Der Typ des Knotens kann über die Eigenschaft ID ermittelt bzw. festgelegt werden:

- _XmlNodeDocument (7) - Dokument

Alle Knoten des Dokuments sind diesem Knoten untergeordnet. Der Wurzelknoten enthält ein Attribut zur verwendeten XML-Version. Das Attribut zum verwendeten Zeichensatz wird bei der Anweisung XmlSave() angegeben. Zusätzlich kann das Attribut "standalone" auf "yes" oder "no" gesetzt werden: tNodeDoc->CteInsertNode('standalone',
_XmlNodeAttribute, 'yes', _CteAttrib)

- _XmlNodeElement (8) - Element

Kontakt

Der Name des Elements befindet sich in der Eigenschaft Name, als Wert wird NULL angegeben.

- _XmlNodeAttribute (9) - Attribut

Der Name des Attributs befindet sich in der Eigenschaft Name, der Wert in der Eigenschaft ValueAlpha.

- _XmlNodeComment (10) - Kommentar

Der Kommentar befindet sich in der Eigenschaft ValueAlpha. Angaben in der Eigenschaft Name werden ignoriert.

- _XmlNodeText (11) - Text

Der Text befindet sich in der Eigenschaft ValueAlpha. Angaben in der Eigenschaft Name werden ignoriert.

- _XmlNodeCDATA (12) - Daten

Die Daten befindet sich in der Eigenschaft ValueAlpha. Angaben in der Eigenschaft Name werden ignoriert.

- _XmlNodeProcessingInstruction (13) - Verarbeitungsanweisung

Das Ziel der Anweisung befindet sich in der Eigenschaft Name, die Daten in der Eigenschaft ValueAlpha.

- _XmlNodeDocumentType (14) - Dokumenttyp

Der Name des Wurzelelements befindet sich in der Eigenschaft Name, die Definition in der Eigenschaft ValueAlpha.

Beispiel zum Erstellen einer XML-Datei

```
main local {    tDoc    : handle;    tRoot    : handle;    tParent : handle;  }{  // Create docu
  // attributes to element  tParent->CteInsertNode('state', _XmlNodeAttribute, 'Hessen', _CteAttr
```

Das Beispiel schreibt folgende Datei:

```
<?xml version="1.0" encoding="UTF-8"?><?xmlstylesheet type="text/xsl" href="style.xsl"?><!--List
```

Wird eine XML-Datei aufbereitet, darf in den Value...-Eigenschaften nur Zeichenketten übergeben werden. Andere Werte sind nicht zulässig. Die Zeichenketten dürfen keine Zeichen enthalten, die als Start- oder Ende-Zeichen (zum Beispiel < oder >) verwendet werden. Solche Zeichen müssen mit ihren Zeichencode oder als Zeichenreferenzen (< oder >) angegeben werden. Enthalten Zeichenketten das Start- und Ende-Zeichen, können sie in CDATA-Bereichen (Knotentyp XmlNodeCDATA) abgelegt werden. Zeichenketten in diesen Bereichen dürfen das Ende-Zeichen des CDATA-Bereichs (]) nicht beinhalten.

Befehle

- XmlClose
- XmlError
- XmlGetValueAlpha
- XmlGetIntValue

- [XmlLoad](#)
- [XmlOpenReader](#)
- [XmlOpenWriter](#)
- [XmlRead](#)
- [XmlSave](#)
- [XmlWrite](#)

Konstanten

- [XmlNodeDocumentType](#)
- [XmlNodeElement](#)
- [XmlNodeText](#)
- [XmlNodeAttribute](#)
- [XmlNodeComment](#)
- [XmlNodeProcessingInstruction](#)
- [XmlNodeDefault](#)
- [XmlNodeDTDValidate](#)
- [XmlNodeDefault](#)
- [XmlNodeOverwrite](#)
- [XmlNodePure](#)
- [XmlNodeAttribNode](#)
- [XmlNodeTypeAttribute](#)
- [XmlNodeTypeCDATA](#)
- [XmlNodeTypeComment](#)
- [XmlNodeTypeDocument](#)
- [XmlNodeTypeDocumentFragment](#)
- [XmlNodeTypeDocumentType](#)
- [XmlNodeTypeElement](#)
- [XmlNodeTypeEndElement](#)
- [XmlNodeTypeEntity](#)
- [XmlNodeTypeEntityReference](#)
- [XmlNodeTypeNotation](#)

Kontakt

- [XmlReaderTypePI](#)
- [XmlReaderTypeSigWhitespace](#)
- [XmlReaderTypeText](#)
- [XmlReaderTypeWhitespace](#)
- [XmlSaveDefault](#)
- [XmlSavePure](#)
- [XmlStartDocument](#)
- [XmlStartElement](#)
- [XmlWriteAttribute](#)
- [XmlWriteCDATA](#)
- [XmlWriteComment](#)
- [XmlWriteDocType](#)
- [XmlWriteElement](#)
- [XmlWritePI](#)
- [XmlWriteText](#)

Kontakt

obj -> XmlClose()  Schließt ein XmlReader- hzw.

XmlWriter-Objekt

Deskriptor des

obj XmlReader- bzw.

XmlWriter-Objektes

Verwandte Befehle,

Siehe [XmlOpenReader\(\)](#),

[XmlOpenWriter\(\)](#)

Diese Anweisung schließt das mit [XmlOpenReader\(\)](#) bzw. [XmlOpenWriter\(\)](#) geöffnetes XML-Objekt (obj).

Beispiel:

```
// XmlReader öffnetXmlReader # XmlOpenReader('C:\File.xml', 'C:\Schema.xml');if (tXmlReader > 0)
```

Mögliche Laufzeitfehler:

ErrHdIInvalid Deskriptor (obj) ist kein gültiger XmlReader- oder
XmlWriter-Deskriptor.

Kontakt

XmlError(int1) : alpha  **Weitere Informationen zu einer XML-Fehler ermitteln**

int1 Information, die ermittelt werden soll Resultat

alpha Zeichenkette mit Informationen Siehe

Verwandte Befehle, Blog

Tritt beim Laden einer XML-Datei (**XmlLoad()**) ein Fehler auf, können mit dieser Anweisung weitere Informationen über diesen Fehler ermittelt werden. Folgende Informationen stehen über einen XML-Fehler zur Verfügung:

- **_XmlErrorText (0)**

Es wird der Fehlertext zurückgegeben.

- **_XmlErrorCode (1)**

Es wird der Fehlerwert zurückgegeben.

- **_XmlErrorLine (2)**

Es wird die Zeile, in der der Fehler aufgetreten ist, zurückgegeben.

- **_XmlErrorColumn (3)**

Es wird die Spalte, in der der Fehler aufgetreten ist, zurückgegeben.

Mögliche Laufzeitfehler

ErrValueInvalid In (int1) wurde ein ungültiger Wert übergeben.

Kontakt



obj -> XmlGetValueAlpha(int1) : alpha

Ermittelt eine alphanumerische Eigenschaft eines XML-Knotens oder einer -Datei

obj Deskriptor des XmlReader-Objektes

Zu ermittelnder Wert

_XmlVersion XML-Version

int1 _XmlGetEncoding Kodierung

_XmlGetName Name des Knotens

_XmlGetValue Inhalt des Knotens

Resultat alpha Wert abhängig von der angegebenen Option oder ein Leerstring bei einem Fehler.

Siehe [Verwandte Befehle](#), [XmlRead\(\)](#), [XmlGetInt\(\)](#)

Dieser Befehl liefert Informationen zu einem mit [XmlRead\(\)](#) gelesenen Knoten in der XML-Datei (obj).

Je nach Option (int1) können andere Werte zurückgegeben werden:

Option Beschreibung

_XmlGetVersion (1) XML-Version - Dieser Wert steht nach dem ersten [XmlRead\(\)](#) zur Verfügung.

_XmlGetEncoding Kodierung - Dieser Wert steht nach dem ersten [XmlRead\(\)](#) zur Verfügung.

(2) Verfügbar.

_XmlGetName (3) Name des Knotens

_XmlGetValue (4) Inhalt des Knotens



Die Zeichenketten sind UTF-8 kodiert.

Ist ein Fehler aufgetreten wird eine leere Zeichenkette zurückgegeben.

Beispiel:

```
// XmlReader öffnetXmlReader # XmlOpenReader('C:\File.xml', 'C:\Schema.xml');if (tXmlReader > 0)
    case _XmlReaderTypeElement :           {           tName  # tXmlReader->XmlGetValueAlpha(_XmlGetNam
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor (obj) ist kein XmlReader-Deskriptor.

ErrValueInvalid Die angegebene Option (int1) ist ungültig.

ErrValueRange Ermittelter Wert ist länger als 65.520 Zeichen.

Kontakt



obj -> XmlGetValueInt(int1) : int

Ermittelt eine numerische Eigenschaft eines XML-Knotens oder einer -Datei

obj Deskriptor des XmlReader-Objektes

Zu ermittelnder Wert

_XmlIsStandalone Ist XML-Dokument Standalone

_XmlGetType Typ des Knotens

int1

_XmlNodeDepth Ebenentiefe des Knotens

_XmlIsValid Ist XML-Dokument valide

_XmlGetAttribCount Anzahl der Attribute des Knotens

int Wert abhängig von der angegebenen Option oder **_ErrGeneric** bei

Resultat einem Fehler.

Siehe [Verwandte Befehle](#), [XmlRead\(\)](#), [XmlGetValueAlpha\(\)](#)

Dieser Befehl liefert Informationen zu einem mit [XmlRead\(\)](#) gelesenen Knoten in der XML-Datei

(obj).



Je nach Option (int1) können andere Werte zurückgegeben werden:

Option Beschreibung

_XmlIsStandalone (1) Standalone?

1 yes

0 no

-1 nicht definiert oder Fehler

Dieser Wert steht nach dem ersten [XmlRead\(\)](#) zur

Verfügung.

_XmlGetType (2) Typ des Knotens

_XmlReaderTypeElement (1) Element

_XmlReaderTypeAttribute (2) Attribut

_XmlReaderTypeText (3) Text

_XmlReaderTypeCDATA (4) Daten

_XmlReaderTypeEntityReference (5) Verweis auf

Textbaustein

Textbaustein

_XmlReaderTypeEntity (6) Verarbeitungsanweisung

_XmlReaderTypePI (7) Kommentar

_XmlReaderTypeComment (8) Dokument

_XmlReaderTypeDocument (9) Dokumenttyp

_XmlReaderTypeDocumentFragment (10) Dokumentfragment

(11)

Notation

_XmlReaderTypeWhitespace (12) Leerzeichen

_XmlReaderTypeSigWhitespace (13) Leerzeichen

_XmlReaderTypeEndElement (14) Elementende

_XmlReaderTypeEndElement (15)

_XmlNodeDepth (3) Ebenentiefe des Knotens

Kontakt

_XmlIsValid (7)

Wurde Knoten validiert?

1 Ja

0 Nein



Liefert nur 1, wenn bei XmlOpenReader() die Option XmlOpenReaderDTDValidate angegeben wurde und das Dokument bis zum aktuell gelesenen Knoten valide ist.

_XmlGetAttribCount (8)

Anzahl der Attribute des Knotens

Ist ein Fehler aufgetreten wird ErrGeneric zurückgegeben.

Beispiel:

```
// XmlReader öffnetXmlReader # XmlOpenReader('C:\File.xml', 'C:\Schema.xml');if (tXmlReader > 0)
```

Die Typen, die mit _XmlGetType ermittelt werden sind in folgendem Beispieldokument farblich gekennzeichnet:

- _XmlReaderTypeElement
- _XmlReaderTypeAttribute
- _XmlReaderTypeText
- _XmlReaderTypeCDATA
- _XmlReaderTypeEntityReference
- _XmlReaderTypeEntity
- _XmlReaderTypePI
- _XmlReaderTypeComment
- _XmlReaderTypeDocument
- _XmlReaderTypeDocumentType
- _XmlReaderTypeDocumentFragment
- _XmlReaderTypeNotation
- _XmlReaderTypeWhitespace
- _XmlReaderTypeSigWhitespace
- _XmlReaderTypeEndElement

```
<?xml version="1.0" encoding="utf-8"?><?xml-stylesheet type="text/xsl" href="style.xsl"?><!DOCTYPE
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor (obj) ist kein XmlReader-Deskriptor.

ErrValueInvalid Die angegebene Option (int1) ist ungültig.

Kontakt

obj ->

XmlLoad(alpha1[,
int2[, handle3[,
alpha4[, handle5]]]]):
int
XML-Daten lesen
 Deskriptor eines
 obj CteNode-Objekts
alpha1 Pfad- und Dateiname
 Optionen (optional)
 XmlLoadHugeTextNode Knoten
 mit
 Inhalten
int2 größer
 als 10
 MB
 laden

handle3 Deskriptor eines
 Memory-Objekts (optional)

alpha4 Pfad- und Dateiname einer
 Schema-Datei (optional)
 Deskriptor eines

handle5 Memory-Objekts mit dem
 Schema (optional)

Resultat int Fehlerwert
 Verwandte Befehle, XmlSave(),
 Siehe Beispiel - Durchsuchen einer
 XML-Struktur, Blog

Diese Anweisung lädt XML-Daten aus der in (alpha1) angegebenen externen Datei und erzeugt eine CteNode-Struktur unterhalb des in (obj) übergebenen Objekts. Das übergebene Objekt muss vom Typ CteNode sein.

Optional kann in (int2) folgende Option angegeben werden:

XmlLoadHugeTextNode Knoten mit Inhalten größer als 10 MB laden

Befinden sich die Daten bereits in einem Memory-Objekt, kann das Objekt in (handle3) übergeben werden. In diesem Fall wird die Angabe des Dateinamens in (alpha1) ignoriert.

In (alpha4) bzw. (int5) kann ein Schema als externe Datei bzw. Memory-Objekt angegeben werden.

Als Resultat wird ein Fehlerwert zurückgegeben. Folgende Fehlerwerte können zurückgegeben werden:

- ErrOk
- ErrXmlWarning
- ErrXmlRecoverable
- ErrXmlFatal
- ErrFsi...

Kontakt

Wurde die Anweisung erfolgreich ausgeführt, befinden sich die Informationen in CteNode-Objekten unterhalb des in (obj) übergebenen Objekts und können mit den Befehlen für dynamische Strukturen verarbeitet werden.

Weitere Informationen zu einem XML-Fehler können mit der Anweisung XmleError() ermittelt werden.

Beispiel:

```
sub ReadXML( aHdlMem : handle;) local { tXMLDoc : handle; tXMLAttr  
    for tXMLNode # tXMLItemRoot->CteRead(_CteChildList | _CteFirst); loop tXMLNode  
    // Fehler bei XMLLoad() if (tErr != _ErrOk) { // XML-Fehler if (tErr <= _ErrXmlWarning
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Einer der übergebenen Deskriptoren ist ungültig

ErrValueInvalid In (int2) ist nicht 0 angegeben.

Kontakt

XmlOpenReader(alpha1[, alpha2[, int3]]) :



handle

XML-Datei zum sequenziellen Lesen öffnen

alpha1 Pfad und Name der XML-Datei

alpha2 Schema-Datei (optional)

Optionen (optional)

int3 _XmlOpenReaderDefault Normal öffnen

_XmlOpenReaderDTDValidate Dokumenttypdefinitionen

überprüfen

Deskriptor des XmlReaders oder



Resultat handle Fehlerwert

Siehe Verwandte Befehle, XmlRead(), XmlClose(),

XmlOpenWriter()

Diese Anweisung öffnet die XML-Datei (alpha1) zum sequenziellen Lesen. Ist eine Schema-Datei (alpha2) angegeben, wird die XML-Datei anhand des Schemas überprüft. Das

Überprüfungsergebnis kann nach jedem gelesenen Knoten oder am Ende der Datei mit

XmlGetValueInt(tXmlReader, _XmlIsValid) abgefragt werden.

Folgende Optionen (int3) können angegeben werden:

_XmlOpenReaderDefault (0) Datei ohne Prüfung öffnen (Standard)

_XmlOpenReaderDTDValidate Dokumenttypdefinitionen beim Lesen der Datei

(1) überprüfen

Als Resultat wird ein XmlReader-Objekt oder ein Fehlerwert zurückgegeben. Folgende Fehlerwerte können zurückgegeben werden:

ErrGeneric Allgemeiner Fehler

ErrOutOfMemory Speicher konnte nicht angefordert werden

ErrFsi... Fehler beim Dateizugriff

Beispiel:

```
// XmlReader öffnetXmlReader # XmlOpenReader('C:\File.xml', 'C:\Schema.xml', _XmlOpenReaderDTDVa
```

Mögliche Laufzeitfehler:

ErrValueInvalid Die angegebene Option (int3) ist ungültig.

Kontakt

XmlOpenWriter(alpha1[, int2]) : handle        **XML-Datei**

zum sequenziellen Schreiben öffnen

alpha1 Pfad und Name der XML-Datei

Optionen (optional)

_XmlOpenWriterDefault Datei formatiert schreiben

int2 **_XmlOpenWriterPure** Datei unformatiert schreiben

_XmlOpenWriterOverwrite Existierende Datei überschreiben

Resultat **handle** Deskriptor des XmlWriters oder Fehlerwert 

Verwandte Befehle, [XmlWrite\(\)](#), [XmlClose\(\)](#),

Siehe

[XmlOpenReader\(\)](#)

Diese Anweisung öffnet die XML-Datei (alpha1) zum sequenziellen Schreiben.

Folgende Optionen (int2) können angegeben werden:

_XmlOpenWriterDefault (0) Datei formatiert schreiben (Standard)

_XmlOpenWriterPure (1) Datei unformatiert schreiben

_XmlOpenWriterOverwrite (2) Datei überschreiben, falls sie bereits existiert

Die Optionen (int2) können kombiniert werden.

Als Resultat wird ein XmlWriter-Objekt oder ein Fehlerwert zurückgegeben. Folgende Fehlerwerte können zurückgegeben werden:

ErrGeneric Allgemeiner Fehler

ErrOutOfMemory Speicher konnte nicht angefordert werden

ErrFsi... Fehler beim Dateizugriff

Beispiel:

```
// XmlWriter öffnetXmlWriter # XmlOpenWriter('C:\File.xml', _XmlOpenWriterPure | _XmlOpenWriterO
```

Mögliche Laufzeitfehler:

ErrValueInvalid Die angegebene Option (int2) ist ungültig.

Kontakt

obj -> XmlRead([int1]) : int | Liest den nächsten

Knoten einer XML-Datei

obj Deskriptor des XmlReader-Objektes

Optionen (optional)

int1 _XmlReadAttribNode Liest den nächsten

Attributknoten

Leseresultat:

ErrOk Nächster Knoten gelesen

Resultat int ErrEndOfData Kein nächster Knoten



vorhanden

ErrGeneric Fehler bei der Verarbeitung

Verwandte Befehle, XmlOpenReader(),

Siehe

XmlGetInt(), XmlGetValueAlpha()

Dieser Befehl liest den nächsten Knoten in der XML-Datei (obj). Wird als Option (int1)

_XmlReadAttribNode angegeben, wird von dem Knoten ein Attributknoten gelesen. Andernfalls wird ein normaler Knoten gelesen.

Als Resultat wird ein Fehlerwert zurückgegeben. Folgende Fehlerwerte können zurückgegeben werden:

ErrOk Nächster Knoten wurde erfolgreich gelesen.

ErrEndOfData Es gibt keinen folgenden Hauptknoten oder der Knoten hat keinen Attributknoten (bei Option (int1) = _XmlReadAttribNode).

ErrGeneric Bei der Verarbeitung ist ein Fehler aufgetreten.

Beispiel:

```
// XmlReader öffnetXmlReader # XmlOpenReader('C:\File.xml', 'C:\Schema.xml');if (tXmlReader > 0)
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor (obj) ist kein gültiger XmlReader-Deskriptor.

ErrValueInvalid Die angegebene Option (int1) ist ungültig.

Kontakt

obj ->

XmlSave(alpha1[,
int2[, handle3[, int4[,
int5]]]])) : int

XML-Daten schreiben

Deskriptor des zu
obj schreibenden Objekts

alpha1 Pfad- und Dateiname

Optionen (optional)

_XmlSaveDefault Formatierte

int2 Ausgabe

_XmlSavePure Unformatierte

Ausgabe

Deskriptor eines

handle3 Memory-Objekts (optional)

int4 Zielzeichensatz (optional)

int5 Quellzeichensatz (optional)

Resultat int Fehlerwert

Verwandte Befehle,

Siehe XmlLoad(), Blog



Der Befehl erzeugt aus dem übergebenen Objekt (obj) eine XML-Datei (alpha1) oder schreibt XML in das in (handle3) angegebene Memory-Objekt. Das übergebene Objekt muss vom Typ CteNode, ein Dialog, ein Menü oder ein Druck-Objekt sein. Es kann auch ein mit StoOpen() geöffnetes Objekt übergeben werden. Importierte Kachel-, Raster- und Vektorgrafiken (siehe Ressource importieren) können mit StoExport() und StoReadMem() exportiert werden. Ist die in (alpha1) angegebene Datei bereits vorhanden, wird sie überschrieben.

Als Option kann _XmlSaveDefault (0) oder _XmlSavePure (1) angegeben werden.

Standardmäßig erfolgt eine formatierte (mit Leerzeichen und Zeilenwechsel versehene)

Ausgabe (int2 = _XmlSaveDefault). Mit _XmlSavePure wird keine Formatierung durchgeführt.

Wird in (handle3) ein Memory-Objekt angegeben, wird der Dateiname ignoriert und der Inhalt in das Objekt geschrieben. Der Inhalt wird an den bestehenden Inhalt angehängt. Sofern das Memory-Objekt schon Daten enthält, muss seine in (int4) angegebene Zeichenkodierung mit der bereits existierenden (Eigenschaft Charset) übereinstimmen. Ist dies nicht der Fall wird der Laufzeitfehler ErrHdlInvalid erzeugt.

Bei der Angabe einer Zeichencodierung werden alle Zeichenketten in den Zeichencode gewandelt. Zusätzlich wird in dem Knoten des Dokuments (ID = XmlNodeDocument) das Attribut "encoding" auf die entsprechende Zeichencodierung gesetzt.

Wird in (int5) ein Quellzeichensatz angegeben, wird der Inhalt in von diesem in den in (int4) angegebenen Zielzeichensatz oder den CONZEPT 16-Zeichensatz konvertiert. Ist kein Quellzeichensatz angegeben, wird der CONZEPT 16-Zeichensatz als Quellzeichensatz verwendet.

Kontakt

Konnte die externe Datei nicht geschrieben werden, wird ein ErrFsi...-Fehlerwert zurückgegeben.

Beispiel:

```
sub WriteXML( aHdlMem : handle;) local { tXMLDoc : handle; tXMLItemRoo
// Kommentar tXMLItemRoot->CteInsertNode('' , _XmlNodeComment, 'comment'); // Text tXMLItemRo
```

Das Resultat sieht wie folgt aus:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><!DOCTYPE myRootElement SYSTEM "myDTD.dtd"
```

Mögliche Laufzeitfehler:

- ErrHdlInvalid** Einer der übergebenen Deskriptoren ist ungültig.
In (int2) ist ein ungültiger Wert oder das angegebene
- ErrValueInvalid** Memory-Objekt verfügt über einen anderen Zeichensatz als in (int4) angegeben.

Kontakt

obj -> XmlWrite(int1[, alpha2[, alpha3[,
alpha4[, alpha5[, alpha6]]]]]) : int Schreibt
einen Knoten einer XML-Datei



obj Deskriptor des XmlWriter-Objektes

Typ

_XmlStartDocument Start des

XML-Dokumentes

_XmlEndDocument Ende des

XML-Dokumentes

_XmlElement Neues Element öffnen

_EndElement Element schließen

int1 _XmlWriteElement Neues Element einfügen

_XmlAttribute Attribut schreiben

_XmlWriteDocType DTD-Verweis einfügen

_XmlWritePI Verarbeitungsanweisung
einfügen

_XmlWriteText Text einfügen

_XmlWriteCDATA Daten einfügen

_XmlWriteComment Kommentar einfügen

alpha2 Argument1 (optional)

alpha3 Argument2 (optional)

alpha4 Argument3 (optional)

alpha5 Argument4 (optional)

alpha6 Argument5 (optional)

Fehlercode

Resultat int ErrOk Kein Fehler
aufgetreten



ErrGeneric Fehler aufgetreten

Siehe [Verwandte Befehle, XmlOpenWriter\(\)](#)

Dieser Befehl schreibt einen Knoten in die XML-Datei (obj). Der Inhalt wird vor dem Schreiben gepuffert. Wird die Größe des Puffers überschritten oder XmlWrite(_XmlEndDocument) aufgerufen, wird der Pufferinhalt in die Datei geschrieben. Je nach Typ (int1) müssen die Argumente Argument1 (alpha2) bis Argument5 (alpha6) angegeben werden. Folgende Konstanten können als Typ (int1) angegeben werden:

Typ Beschreibung

_XmlStartDocument (1) Start des XML-Dokumentes

Argument1 Version

Argument2 Zeichensatzkodierung

Argument3 Standalone-Flag

_XmlEndDocument (2) Ende des XML-Dokumentes

Keine Argumente benötigt

Alle offenen Knoten werden geschlossen.

_XmlElement (3) Neues Element öffnen

Kontakt

	Argument1 Name Element
<u>_EndElement (4)</u>	schließen Keine Argumente
	benötigt Neues Element
<u>_XmlElement (5)</u>	einfügen Argument1 Name
	Argument2 Inhalt
	Das Element kann keine Kindelemente oder Attribute enthalten.
	 Attribut schreiben
<u>_XmlAttribute (6)</u>	Argument1 Name
	Argument2 Wert DTD-
	Verweis einfügen
<u>_XmlWriteDocType (7)</u>	Argument1 Name
	Argument2 Speicherort einer Public-DTD
	Argument3 Speicherort einer System-DTD
	Verarbeitungsanweisung einfügen Argument1 Ziel
<u>_XmlWritePI (8)</u>	Argument2 Inhalt Text
	einfügen Argument1 Text
<u>_XmlWriteText (9)</u>	Daten einfügen Argument1
	Daten Kommentar einfügen
<u>_XmlWriteCDATA (10)</u>	Argument1 Kommentar
<u>_XmlWriteComment (11)</u>	Die Zeichenketten werden im CONZEPT 16-Zeichensatz erwartet. Die Ausgabe erfolgt in der _XmlStartDocument unter Zeichensatzkodierung angegebenen Kodierung.

Wurde der Knoten erfolgreich geschrieben, wird ErrOk zurückgegeben, andernfalls ErrGeneric.

Beispiel:

```
// XmlWriter öffnetXmlWriter # XmlOpenWriter('C:\XML.xml', _XmlOpenWriterDefault);if (tXmlWriter  
    tXmlWriter->XmlWrite(_XmlWriteComment, 'Kunde 4711');           tXmlWriter->XmlWrite(_XmlSta
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der Deskriptor (obj) ist kein gültiger XmlWriter-Deskriptor.

ErrValueInvalid Der angegebene Typ (int1) ist ungültig.

ErrNoArgument Mindestens eines der benötigten Argumente (alpha2 - alpha6) fehlt.

JSON-Verarbeitung

Verarbeitung von JSON in CONZEPT 16

Befehlsgruppen,
Befehlsliste,

Siehe JsonLoad(),
JsonSave(),
Beispiel, Blog

Das JSON-Format (JavaScript Object Notation) ist ein Menschen lesbaren Format. Es ist wesentlich einfacher als XML und orientiert sich an Standards von Computersprachen. Nähere Informationen befinden sich auf der Seite www.json.org/.

JSON-Dateien besitzen eine hierarchische Struktur. Diese wird mit CteNode-Objekten nachgebildet. Die Informationen innerhalb der Datei sind in *name : value*-Paaren abgelegt. Jedes dieser Paare wird in einem Knoten gespeichert. Der Typ des Knotens und des gespeicherten Wertes kann über die Eigenschaft ID ermittelt und mit folgenden Konstanten verglichen werden:

- _JsonNodeArray (1) - Array

In den untergeordneten Objekten zu diesem Knoten sind die Elemente der Liste enthalten.

- _JsonNodeObject (2) - Objekt

In den untergeordneten Objekten sind die Wert-Paare enthalten. Der Name steht dabei in der Eigenschaft Name. Der Datentyp des Wertes steht in der Eigenschaft Type. Der Typ kann mit den _Type...-Konstanten verglichen werden. Der Wert steht in der entsprechenden Value...-Eigenschaft.

- _JsonNodeString (3) - alpha

Bei dem Wert handelt es sich um eine Zeichenkette. Die Zeichenkette steht in der Eigenschaft ValueAlpha.

- _JsonNodeNumber (4) - int, bignum, float, decimal

Bei dem Wert handelt es sich um eine Nummer. Der Datentyp kann über die Eigenschaft Type ermittelt werden. Der Wert steht in der entsprechenden Value...-Eigenschaft.

- _JsonNodeBoolean (5) - logic

Bei dem Wert ist entweder true oder false angegeben. Der Wert steht in der Eigenschaft ValueLogic.

- _JsonNodeNull (6) - NULL Der

Wert ist nicht angegeben.

Nach dem Einlesen einer JSON-Datei mit der Anweisung JsonLoad() kann auf die Elemente entweder über die hierarchische Struktur oder direkt über den Namen zugegriffen werden. Wird ein Element mit einem bestimmten Namen gesucht, empfiehlt sich der Zugriff über den Namen.

Kontakt

Befehle für JSON-Verarbeitung

Liste der Befehle und Konstanten zur Verarbeitung von JSON

Befehlsgruppen,

Siehe Befehlsliste,

JSON

Befehle

- JsonLoad
- JsonSave

Konstanten

- JsonNodeArray
- JsonNodeBoolean
- JsonNodeNull
- JsonNodeNumber
- JsonNodeObject
- JsonNodeString
- JsonSaveDefault
- JsonSavePure

Kontakt

obj ->
JsonLoad(alpha1[,
int2[, handle3]]) :
int
JSON-Daten lesen
 Deskriptor eines
obj CteNode-Objekts
alpha1 Pfad- und Dateiname
int2 0 - Reserviert (optional)
 Deskriptor eines
handle3 Memory-Objekts
 (optional)
 ErrOk oder
Resultat int
 Fehlerposition



Siehe [JSON](#), [JsonSave\(\)](#),

[Beispiel](#), [Blog](#)

Der Befehl lädt JSON-Daten aus der Datei (alpha1) oder dem **Memory**-Objekt (handle3) und erzeugt eine **CteNode-Struktur** unterhalb des angegebenen Objekts (obj). Das Objekt muss vom Typ **CteNode** sein.

Informationen über das JSON-Format (JavaScript Object Notation) befinden sich auf www.json.org/.

Wird in (handle3) ein Objekt übergeben, wird der Dateiname (alpha1) ignoriert.

Der Übergabeparameter (int2) ist reserviert und muss mit 0 übergeben werden.

Der Befehl liefert **ErrOk**, wenn kein Fehler auftrat. Tritt beim Lesen der Datei ein Fehler auf, wird entweder ein entsprechender Wert (**ErrFsi...**) oder die Fehlerposition innerhalb der Datei zurückgegeben (Rückgabewert > 0).

Beispiel:

```
sub JSONLoadFromFile() local { tCteNodeJSON : handle; tErr : int; }{ tCteNodeJS
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Einer der übergebenen Deskriptoren ist ungültig.

ErrValueInvalid In (int2) wurde ein Wert ungleich 0 übergeben.

Kontakt

obj ->

JsonSave(alpha1[,
int2[, handle3[, int4[,
int5]]]]) : int



JSON-Daten schreiben
Deskriptor eines
obj

CteNode-Objekts

alpha1 Pfad- und Dateiname

Optionen (optional)

_JsonSaveDefault Formatierte

int2 Ausgabe

_JsonSavePure Unformatierte

Ausgabe

Deskriptor eines

handle3 Memory-Objekts (optional)

int4 Zielzeichensatz (optional)

int5 Quellzeichensatz (optional)

Resultat int Fehlerwert



Verwandte Befehle, JSON,

Siehe

JsonLoad(), Beispiel, Blog

Der Befehl erzeugt aus dem übergebenen Objekt (obj) eine JSON-Datei (alpha1) oder schreibt JSON in das in (handle3) angegebene Memory-Objekt. Das übergeben Objekt muss vom Typ CteNode sein. Ist die in (alpha1) angegebene Datei bereits vorhanden, wird sie überschrieben.

Als Option kann _JsonSaveDefault (0) oder _JsonSavePure angegeben werden.

Standardmäßig erfolgt eine formatierte (mit Leerzeichen und Zeilenwechsel versehende) Ausgabe (int2 = _JsonSaveDefault). Mit _JsonSavePure wird keine Formatierung durchgeführt.

Wird in (handle3) ein Memory-Objekt angegeben, wird der Dateiname ignoriert und der Inhalt in das Objekt geschrieben. Der Inhalt wird an den bestehenden Inhalt angehängt. Sofern das Memory-Objekt schon Daten enthält, muss seine in (int4) angegebene Zeichenkodierung mit der bereits existierenden (Eigenschaft Charset) übereinstimmen. Ist dies nicht der Fall wird der Laufzeitfehler ErrHdlInvalid erzeugt.

Wird in (int5) ein Quellzeichensatz angegeben, wird der Inhalt von diesem in den, in (int4) angegebenen Zielzeichensatz, oder in den CONZEPT 16-Zeichensatz konvertiert. Ist kein Quellzeichensatz angegeben, wird der CONZEPT 16-Zeichensatz als Quellzeichensatz verwendet.

Tritt bei der Erzeugung der externen Datei ein Fehler auf, wird ein entsprechender Wert (ErrFsi...) zurückgegeben. Tritt kein Fehler auf, ist der Rückgabewert ErrOk.

Beispiel:

```
sub JSONSaveToFile() local { tCteNodeJSON : handle; tCteNodeJSONItem : handle; tErr #  
    tCteNodeJSON->JsonSave(_Sys->spPathMyDocuments + '\JSON-data.txt'); // Cte-Knoten leere
```

Das Resultat sieht wie folgt aus:

Kontakt

```
{  "object": { "one": "1", "two": 2 },  "array": [ "1", 2 ],  "number": 1234567890,  "bo
```

Mögliche Laufzeitfehler:

- ErrHdlInvalid** Einer der übergebenen Deskriptoren ist ungültig.
In (int2) ist ein ungültiger Wert oder das angegebene
ErrValueInvalid Memory-Objekt verfügt über einen anderen Zeichensatz als in (int4)
angegeben.

Kontakt

COM-Befehle

Befehle zur Kommunikation mit der COM-Schnittstelle

[Beispiel](#), [Liste](#)

[sortiert nach](#)

[Gruppen](#),

[Alphabetische](#)

[Liste aller](#)

Siehe

Befehle

- [ComArgGet](#)
- [ComArgSet](#)
- [ComCall](#)
- [ComCallResult](#)
- [ComClose](#)
- [ComInfo](#)
- [ComOpen](#)
- [ComPropGet](#)
- [ComPropGetText](#)
- [ComPropSet](#)
- [ComPropSetText](#)

Konstanten

- [ComAppCreate](#)
- [ComInfoErrCode](#)
- [ComInfoErrText](#)

Das Component Object Model (COM) ist eine Software Architektur, die es ermöglicht, über eine binäre Schnittstelle auf Anwendungen und Komponenten von verschiedenen Herstellern zuzugreifen. Es beruht auf einem objektbasierenden Modell, das zwischen Schnittstelle und Implementierung trennt. Die Funktionalität wird über eine Sammlung von Objekten angesprochen. Diese Objekte bestehen aus Eigenschaften, Methoden und Ereignissen. Dieses Objektmodell mit den dazugehörigen Methoden wird in der Dokumentation des entsprechenden Programms beschrieben.



In CONZEPT 16 werden COM-Ereignisse nicht unterstützt. Von dieser Ausnahme sind auch Eigenschaften und Methoden betroffen, die keinen entsprechenden CONZEPT 16-Variablenotyp als Argument erwarten oder als Resultat zurückgeben.

Die Objekte sind hierarchisch aufgebaut. Somit kann ein Objekt eine Sammlung von Unterobjekten beinhalten. Statt dafür jeweils einen Befehl zu implementieren, wird auf diese Objekte über eine Eigenschaft zugegriffen, die als Resultat ein Container-Objekt zurück liefert.

Durch das Auslesen von Eigenschaften können somit der Zustand, Informationen oder wiederum andere Objekte abgefragt werden. Durch das Setzen von Eigenschaften kann das Verhalten der Objekte gesteuert werden. Methoden sind mit Funktionsaufrufen vergleichbar, wobei der Unterschied zum Setzen einer Eigenschaft nur vom Design der Komponente bestimmt ist.

Kontakt

COM-Objekte werden in CONZEPT 16 über Deskriptoren abgebildet. Eine Verbindung zu einem COM-Server wird über den Befehl ComOpen() oder über die COM-Objekte hergestellt. Die Eigenschaften werden über den Befehl ComPropSet() gesetzt und über den Befehl ComPropGet() ausgelesen. Parallel dazu wird auch die Kurzschreibweise über Konstanten unterstützt. Die Konstanten zu Eigenschaften von COM-Objekten setzen sich wie folgt zusammen:

cp<Typ><Eigenschaft>

<Typ> ist eine Abkürzung des Datentypes der Eigenschaft. Folgende Kürzel können angegeben werden:

<Typ> CONZEPT 16-Typ COM Typ

a	<u>alpha</u>	String
c	<u>caltime</u>	variant time
f	<u>float</u>	Gleitkomma
h	<u>handle</u>	Objekt oder Klasse
i	<u>int</u>	Long oder Single
l	<u>logic</u>	Boolean
x	<u>Extended</u>	Array
x?	<u>Extended</u>	Array eines bestimmten Typs (a, c, f, h, i, l)

<Eigenschaft> ist der Name der Eigenschaft eines bestimmten Objekts. Wird eine Eigenschaft gesetzt, die nur gelesen werden kann, erfolgt bei der Verwendung der Konstanten ein Laufzeitfehler. Der Befehl ComPropSet() liefert dagegen einen Wert ungleich 0 als Ergebnis zurück.

```
tComApp # ComOpen('Word.Application', _ComAppCreate); tComApp->cplVisible # true;
```

Äquivalent dazu kann das Setzen der Eigenschaft Visible auch folgendermaßen durchgeführt werden:

```
tComApp->ComPropSet('Visible', true);
```

In einigen Applikationen ist eine Eigenschaft in weitere Eigenschaften unterteilt. Diese Eigenschaften können in der Kurzschreibweise mit cp<Typ><Eigenschaft>(<Eigenschaft>) angesprochen werden.

Im folgenden ein Beispiel zum Lesen und Setzen einer Eigenschaft ("ServerHTTPRequest" vom Typ logic) in einer Eigenschaft ("ClientProperty").

```
tValue # tComObject->cplClientProperty('ServerHTTPRequest'); tComObject->cplClientProperty('Server
```

Existiert eine angegebene Eigenschaft nicht, wird bei Verwendung der Konstanten ein Laufzeitfehler generiert. Werden die Befehle ComPropSet() oder ComPropGet() verwendet, wird der Wert false zurückgegeben.

Erweiterte Eigenschaften (x) können nur über Konstanten angesprochen werden.

Weitere Informationen befinden sich im Abschnitt Erweiterte Eigenschaften.

Kontakt

Methoden werden mit dem Befehl ComCall() aufgerufen und sind immer Bestandteil eines Objekts. Beim Aufruf einer Methode ist zu beachten, dass die Verarbeitung nicht in CONZEPT 16 stattfindet. Je nach Implementierung kann der Aufruf noch vor Beendigung der Methode zurückkehren. Wird zum Beispiel eine länger laufende Berechnung angestoßen oder ein Makro aufgerufen und anschließend sofort auf das Ergebnis zugegriffen, kann es zu einem fehlerhaften Verhalten kommen, wenn die Durchführungszeit nicht abgewartet wird.

Liefert eine aufgerufene Methode ein ergebnis zurück, kann das Ergebnis mit der Anweisung ComCallResult() ermittelt werden.

Im Abschnitt Verwendung der COM-Schnittstelle von Microsoft Word wird die Benutzung der Schnittstelle erläutert.

Kontakt

obj -> ComArgGet(int1,



var2) : int

COM-Argument ermitteln

obj **COM-Argument-Objekt**
 Nummer des

int1 **Arguments**

var2 **Wert**

Resultat int Fehlerwert

Verwandte Befehle,

Siehe ComArgSet(),

EvtCtxEvent

Mit diesem Befehl kann der Wert eines COM-Argumentes ermittelt werden. In (obj) muss ein COM-Argument-Objekt übergeben werden. Dieses wird beim Ereignis EvtCtxEvent im Argument aComArguments übergeben.

Im Parameter (int1) wird das zu lesende Argument definiert. Die Argumente sind von 1 an fortlaufend nummeriert. Im Parameter (var2) wird eine Variable oder ein Feld angegeben, welches nach der Rückkehr der Funktion den Wert des Argumentes enthält. Der Typ von (var2) muss mit dem Typ des COM-Argumentes kompatibel sein.

Die Ereignisse des CtxDocEdit-Objektes und ihre Argumente sind auf der Hersteller-Seite des Moduls beschrieben.

Als Rückgabewert kann der Wert ErrGeneric zurückgegeben werden, wenn ein interner Fehler aufgetreten ist. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.

Beispiel

```
sub EvtCtxEvent( aEvt : event; // Ereignis aEventID : int;
```

Mögliche Laufzeitfehler

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein COM-Argument-Objekt.

ErrValueRange Die Argument-Nummer (int1) ist ungültig oder der Wert (var2) konnte nicht konvertiert werden.

ErrFldType Der Wert (var2) ist nicht kompatibel mit dem des COM-Argumentes.

Kontakt

obj ->
ComArgSet(int1, var2) 
: int
COM-Argument setzen
obj COM-Argument-Objekt
Nummer des
int1 Arguments
var2 Wert
Resultat int Fehlerwert
Verwandte Befehle,
Siehe ComArgGet(),
EvtCtxEvent

Mit diesem Befehl kann der Wert eines COM-Argumentes gesetzt werden. In (obj) muss ein COM-Argument-Objekt übergeben werden. Dieses wird beim Ereignis EvtCtxEvent im Argument aComArguments übergeben.

Im Parameter (int1) wird das zu schreibende Argument definiert. Die Argumente sind von 1 an fortlaufend nummeriert. Im Parameter (var2) wird eine Konstante, eine Variable oder ein Feld angegeben, welches den zu setzenden Wert des COM-Argumentes enthält. Der Typ von (var2) muss mit dem Typ des COM-Argumentes kompatibel sein. Darüber hinaus muss Das COM-Ereignis in diesem Argument eine Rückgabe erwarten.

Die Ereignisse des CtxDocEdit-Objektes und ihre Argumente sind auf der Hersteller-Seite des Moduls beschrieben.

Als Rückgabewert kann der Wert ErrGeneric zurückgegeben werden, wenn ein interner Fehler aufgetreten ist. Bei der Rückgabe von ErrOk ist kein Fehler aufgetreten.

Beispiel

```
sub EvtCtxEvent( aEvt : event; // Ereignis aEventID : int;
```

Mögliche Laufzeitfehler

- ErrHdlInvalid** Bei (obj) handelt es sich nicht um ein COM-Argument-Objekt.
- ErrValueRange** Die Argument-Nummer (int1) ist ungültig oder der Wert (var2) konnte nicht konvertiert werden.
- ErrFldType** Der Wert (var2) ist nicht kompatibel mit dem des COM-Argumentes.

Kontakt

obj -> ComCall(alpha1 [[, var2]])  Aufruf einer

Methode eines COM-Objektes

obj Objekt
alpha1 Name der

 Methode

var2 Parameter

Verwandte

Siehe Befehle,
ComCallResult()

Mit diesem Befehl wird eine Methode eines COM-Objektes aufgerufen. Der Deskriptor des Objekts wird in (obj) übergeben. Der Deskriptor wird entweder durch den Befehl ComOpen(), einem Ctx-Objekt oder durch eine Eigenschaft eines anderen COM-Objekts zurückgegeben.

In (alpha1) steht der Name der Methode.

Sollen der Methode Parameter übergeben werden, müssen diese als weitere Parameter beim Aufruf des Befehls angegeben werden. Eine Überprüfung der Parameter erfolgt erst zur Laufzeit. Es können bis zu 24 Parameter einer Methode übergeben werden. Bei Methoden, die optionale Argumente erwarten, kann mit dem Schlüsselwort NULL ein Argument übersprungen werden. Soll ein Deskriptor als Parameter an die aufgerufene Methode übergeben werden, muss das Schlüsselwort handle dem Parameter vorangestellt werden.

Variablen können auch als var-Parameter übergeben werden (Call-By-Reference). Dem Variablenamen wird dann var vorangestellt. Folgende Datentypen können als var-Parameter übergeben werden:

- logic
- byte
- word
- int
- float



Je nach Implementierung der aufgerufenen Methode kann der Aufruf schon zurückkehren bevor das Ergebnis der Methode zur Verfügung steht.

Das Ergebnis der aufgerufenen Methode kann mit dem Befehl ComCallResult() ermittelt werden.

Kontakt

obj -> ComCallResult(var1)  Ergebnis eines _____

ComCall-Aufrufs ermitteln

obj Objekt

 Variable

var1 beliebigen

 Typs

Verwandte

Siehe Befehle,

ComCall()

Mit dieser Anweisung kann das Ergebnis einer aufgerufenen Methode eines COM-Objekts abgefragt werden. Zuvor muss die Methode mit dem Befehl ComCall() aufgerufen werden. Anschließend kann mit ComCallResult() das Resultat abgefragt werden.

Als (obj) wird das gleiche Objekt wie beim Aufruf von ComCall() angegeben. Abhängig von Rückgabewert der Methode, muss eine Variable (var1) des entsprechenden Typs angegeben werden. Die Variable enthält nach dem Aufruf den Rückgabewert der Methode.

Beispiel:

```
local{ tDefaultTheme : alpha(80); }{ ... tComApp # ComOpen('Word.Application', _ComAppCreate); }
```

Kontakt

obj -> ComClose()  Deskriptor eines COM-

Objektes freigeben

obj Objekt

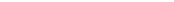
Verwandte

Siehe Befehle,
ComOpen()

Der Befehl gibt den in (obj) übergebenen Deskriptor frei. Der Deskriptor kann entweder über ComOpen() oder eine Eigenschaft eines anderen COM-Objektes erzeugt worden sein.

Jedes Objekt, das erzeugt wurde, sollte mit dieser Funktion wieder entfernt werden.

Kontakt

obj -> ComInfo(int1) : alpha  Informationen zu einem COM-Objekt ermitteln

obj Referenzobjekt oder 0

Optionen

int1 ComInfoErrCode Fehlerwert
 ComInfoErrText Fehlertext

Resultat alpha Fehlerwert oder -text

Siehe [Verwandte Befehle](#)

Mit diesem Befehl können Informationen zu Fehlermeldungen eines COM-Objekts ermittelt werden.

Wurden eine Verbindung zu einem COM-Server mit der Anweisung ComOpen() hergestellt, darf kein Referenz-Objekt übergeben werden. Es wird immer der letzte Fehler ermittelt.

Ein Referenz-Objekt wird bei der Verwendung der Objekte CtxOffice, CtxAdobeReader bzw. WebNavigator benötigt. Wird ein Deskriptor auf ein solches Objekt übergeben, wird der Fehlerwert oder Fehlertext ermittelt, der bei der Initialisierung des Objekts aufgetreten ist.

Folgende Optionen können angegeben werden:

ComInfoErrCode Ermittelt den Fehlerwert aus der Applikation

ComInfoErrText Ermittelt den Fehlertext

Kontakt

ComOpen(alpha1[, int2]) : handle Verbindung



zur COM-Schnittstelle öffnen

alpha1 Application und Objekt

Optionen

int2 ComAppCreate Applikation

starten

Resultat handle COM-Objekt

Verwandte Befehle,

Siehe ComClose(), ComPropGet(),

ComPropSet()

Dieser Befehl öffnet die COM-Schnittstelle zu einem Objekt innerhalb einer Applikation. Der Name der Applikation und des Objektes müssen in (alpha1) durch einen Punkt (.) getrennt angegeben werden.

Über den optionalen Parameter (int2) kann bestimmt werden, ob die Applikation gestartet werden soll (int2 = ComAppCreate). Wird der Parameter nicht angegeben, muss die Applikation bereits gestartet sein, um eine Verbindung aufnehmen zu können.



Eine Applikation, die gestartet wurde, muss mit einer entsprechenden Methode (zum Beispiel "Quit" bei Microsoft Word) beendet werden. Der Befehl ComClose() beendet nicht die Applikation.

Beispiel:

```
// Microsoft Word startetComApp # ComOpen('Word.Application', _ComAppCreate);...// Microsoft Wor
```

Die Konstante wdDoNotSaveChanged hat den Wert 0.

Kontakt

obj -> ComPropGet(alpha1, var2[, int3, ..., int6]) : 
int

Eigenschaft eines COM-Objektes abfragen

obj COM-Objekt

alpha1 Name der
 Eigenschaft
 Wert der

var2 Eigenschaft

int3-int6 zusätzliche Informationen

Resultat int Fehlerwert



Siehe [Verwandte Befehle](#),
[ComOpen\(\)](#),

[ComPropSet\(\)](#),
[ComPropGetText\(\)](#)

Dieser Befehl liest eine Eigenschaft eines COM-Objektes aus.

In (obj) wird der Deskriptor des COM-Objektes und in (alpha1) der Name der auszulesenden Eigenschaft übergeben. Der Wert der Eigenschaft wird in die in (var2) angegebenen Variablen übertragen.

Die Typenüberprüfung findet erst zur Laufzeit statt.

Kann die Eigenschaft nicht gelesen werden, kommt es zu einem Laufzeitfehler. Laufzeitfehler können in einem try-Block abgefangen und verarbeitet werden (siehe auch ErrTryCatch()). Der Rückgabewert entspricht dem Laufzeitfehler oder ErrOk.

Beispiel:

```
local{ tCaption : alpha; }...// Microsoft Word startentComApp # ComOpen('Word.Application', _Co
```

Das Auslesen der Eigenschaft kann ebenfalls über entsprechende Konstanten erfolgen. Die Zusammensetzung der Konstanten ist im Abschnitt COM-Befehle beschrieben.

```
local{ tCaption : alpha; }...// Microsoft Word startentComApp # ComOpen('Word.Application', _Co
```

Muss bei einer Eigenschaft ein Parameter vom Typ Enum angegeben werden, kann der entsprechende Parameter in (int3) angegeben werden. Bei der Verwendung von Konstanten, wird der Parameter in einer Klammer hinter der Konstante angegeben.

```
local{ tCurrencyCode : alpha; }...// Microsoft Word startentComApp # ComOpen('Word.Application'
```

Die Konstante wdCurrencyCode wird mit dem Wert 20 definiert.

Soll die erweiterte Eigenschaft Array ausgelesen werden, muss in den Parametern (int3) - (int6) die Zelle des Array angegeben werden.

```
// Zelle eines 2-dimensionales Arrays abfragentArray->ComPropGet('Item', aValue, 1, 1);
```

Kontakt

obj -> ComPropGetText(alpha1, handle2[, int3, ..., int6]) : int Auslesen

einer alpha-Eigenschaft mit mehr als 8192 Zeichen



obj COM-Objekt

alpha1 Name der Eigenschaft

handle2 Deskriptor des Textpuffers

int3-int6 zusätzliche Informationen

Resultat int Funktion erfolgreich (= ErrOk)
Verwandte Befehle, ComOpen(),



Siehe

ComPropSetText(), ComPropGet()

Der Inhalt der Eigenschaft (alpha1) eines Com-Objektes (obj) kann mit diesem Befehl in einen Textpuffer (handle2) übertragen werden. Der Textpuffer muss zuvor mit dem Befehl TextOpen() angelegt worden sein.

Somit können alpha-Eigenschaften ausgelesen werden, die mehr als 8192 Zeichen beinhalten. Der Rückgabewert gibt an, ob das Auslesen der Eigenschaft erfolgreich war (= ErrOk) oder nicht (!= ErrOk).

Soll die erweiterte Eigenschaft Array ausgelesen werden, muss in den Parametern (int3) - (int6) die Zelle des Arrays (1., 2. und gegebenenfalls 3. Dimension) angegeben werden.

```
// Zelle eines 2-dimensionales Arrays abfragentArray->ComPropGetText('Item', tHd1Text, 1, 1);
```

Kontakt

obj -> ComPropSet(alpha1, var2[, int3,

..., int6]) : int

Eigenschaft eines COM-Objektes setzen

obj COM-Objekt

alpha1 Name der Eigenschaft

var2 Wert der Eigenschaft

int3-int6 zusätzliche Informationen

Resultat int Funktion erfolgreich (= ErrOk)

Siehe Verwandte Befehle, ComOpen(),

ComPropSetText(), ComPropGet()

Dieser Befehl setzt eine Eigenschaft eines COM-Objektes.

In (obj) wird der Deskriptor des COM-Objektes und in (alpha1) der Name der zu setzenden Eigenschaft übergeben. Der Wert der Eigenschaft wird in (var2) angegeben.

Die Typenüberprüfung findet erst zur Laufzeit statt.

Über den Rückgabewert kann überprüft werden, ob das Setzen der Eigenschaft gelungen ist (Rückgabewert = ErrOk) oder nicht.

Beispiel:

```
...// Microsoft Word startentComApp # ComOpen('Word.Application', _ComAppCreate); // Eigenschaft C
```

Das Setzen der Eigenschaft kann ebenfalls über entsprechende Konstanten erfolgen.

Die Zusammensetzung der Konstanten ist im Abschnitt COM-Befehle beschrieben.

Beispiel:

```
...// Microsoft Word startentComApp # ComOpen('Word.Application', _ComAppCreate); // Eigenschaft C
```

Muss bei einer Eigenschaft ein Parameter vom Typ Enum angegeben werden, kann der entsprechende Parameter in (int3) angegeben werden. Bei der Verwendung von Konstanten wird der Parameter in einer Klammer hinter der Konstanten angegeben.

Soll die erweiterte Eigenschaft Array gesetzt werden, muss in den Parametern (int3) - (int6) die Zelle des Arrays angegeben werden.

```
// Zelle eines 2-dimensionales Arrays setzentArray->ComPropSet('Item', 'Jahr', 1, 1);
```



Kontakt

obj -> ComPropSetText(alpha1, handle2) : int



Setzen einer alpha-Eigenschaft mit mehr als 8192 Zeichen
obj COM-Objekt

alpha1 Name der Eigenschaft handle2 Deskriptor
des Textpuffers Resultat int Funktion erfolgreich (=
ErrOk)



Siehe [Verwandte Befehle](#), [ComOpen\(\)](#),
[ComPropSet\(\)](#), [ComPropGetText\(\)](#)

Der Inhalt der Eigenschaft (alpha1) eines Com-Objektes (obj) kann mit diesem Befehl aus einem Textpuffer (handle2) übertragen werden. Der Textpuffer muss zuvor mit dem Befehl [TextOpen\(\)](#) angelegt und mit Inhalt gefüllt worden sein.

Somit können alpha-Eigenschaften gesetzt werden, die mehr als 8192 Zeichen beinhalten. Der Rückgabewert gibt an, ob das Setzen der Eigenschaft erfolgreich war (= ErrOk) oder nicht (!= ErrOk).

Kontakt

Erweiterte Eigenschaften

Beschreibung von Eigenschaften von COM-Objekten

Siehe Befehle der
COM-Schnittstelle

In diesem Abschnitt werden die erweiterten Eigenschaften von COM-Objekten beschrieben. Diese Eigenschaften sind nicht Bestandteil der eingebundenen Anwendung. Sie werden von CONZEPT 16 verwaltet, um die Kommunikation zwischen CONZEPT 16 und der Anwendung zu erleichtern.

Zur Zeit ist die erweiterte Eigenschaften Array realisiert.

Kontakt

Array / cpxArray

**Beschreibung der erweiterten Eigenschaft Array Siehe
Erweiterte Eigenschaften**

Mit dieser Eigenschaft kann ein Array mit bis zu vier Dimensionen angelegt werden. Die Größe des Arrays werden als Parameter der Eigenschaft übergeben. Das Array kann jedem COM-Objekt zugeordnet werden.

Beispiel:

```
tComWorksheet->ComPropGet('Array', tArray, 10, 2);
```

oder alternativ dazu

```
tArray # tComWorksheet->cpxArray(10, 2);
```

In diesem Beispiel wird ein zweidimensionales Array mit zehn Zeilen und zwei Spalten angelegt. Der Typ des Arrays wird hier nicht definiert.

Das Objekt Array verfügt über folgende Eigenschaften:

- **Item**

Mit dieser Eigenschaft wird der Inhalt des Arrays gesetzt bzw. abgefragt. Je nach Anzahl der Dimensionen erwartet die Eigenschaft ein bis vier Parameter. Der Typ der Zelle wird in der Konstante mit angegeben und kann für jede Zelle unterschiedlich sein. Der Typ der Zelle wird durch den dritten Buchstaben der Eigenschaft bestimmt (vgl. COM-Befehle).

Beispiel:

```
tArray->cpaItem(1, 1) # 'Year';tArray->cpaItem(1, 2) # 'Sales';tArray->cpiItem(2, 1) # 199
```

- **ItemType**

In dieser Eigenschaft kann der Typ einer Zelle ermittelt werden. Je nach Anzahl der Dimensionen erwartet die Eigenschaft ein bis vier Parameter. Die Eigenschaft kann nur ausgelesen werden und gibt Werte zurück, die mit den Konstanten für Feld- und Datentypen verglichen werden können.

```
switch (tArray->cpiItemType(tRow, tColumn)){ case _TypeAlpha : ... case _TypeInt : ...}
```

- **Dimension**

Mit dieser Eigenschaft kann die Dimensionalität des Arrays abgefragt werden. In unserem Beispiel ist das Ergebnis 2. Die Eigenschaft kann nicht gesetzt werden.

- **IndexStart**

Mit dieser Eigenschaft kann der Startindex des Arrays ermittelt werden. Für jede Dimension steht diese Eigenschaft zur Verfügung.

- **IndexEnd**

Mit dieser Eigenschaft kann der Endeindex des Arrays ermittelt werden. Für

Kontakt

jede Dimension steht diese Eigenschaft zur Verfügung.

```
for i # 1;loop Inc(i);until (i > tArray->cpiDimension){ tStart # tArray->cpiIndexStart
```

Ein Array wird benötigt, wenn eine Tabelle in Microsoft Excel gefüllt werden soll. Zum Übertragen von Werten zu Excel wird das COM-Objekt Range benötigt.

Beispiel:

```
tComApp # ComOpen('Excel.Application', _ComAppCreate);tComApp->ComCall('Workbooks.Add');tComWorkb
```

Der Bereich A1 bis B10 kann jetzt mit folgender Anweisung gefüllt werden:

```
tComRange->cphValue # tArray;
```

Die Definition des zu füllenden Bereiches kann ebenfalls durch die Angabe der Zellen mit numerischen Werten erfolgen:

```
// Alternative zu tComRange # tComActiveSheet->cphRange('A1:B10');tCellStart # tComWorkSheet->cph
```

Bei der Abfrage der Eigenschaft Range muss handle mit angegeben werden, da hier sonst der übergebene Wert als Zellkoordinate interpretiert wird. Mit handle wird der Wert als Deskriptor gekennzeichnet.

Array mit definiertem Typ

Beim Anlegen eines Arrays mit cpxArray(); kann jedes Element einen anderen Typ haben. In einigen Fällen wird jedoch ein bestimmter Typ benötigt. In diesem Fall kann man das Array mit diesem Typ anlegen. Dazu wird der Typ in der Form cpx<Typ>Array(); angegeben. Die folgenden Typen können angegeben werden:

<Typ> CONZEPT 16-Typ COM Typ

a	<u>alpha</u>	String
c	<u>caltime</u>	variant time
f	<u>float</u>	Gleitkomma
h	<u>handle</u>	Objekt oder Klasse
i	<u>int</u>	Long oder Single
l	<u>logic</u>	Boolean

Kontakt

Beispiel - Verwendung der COM-Schnittstelle von Microsoft Word Einstieg in die Benutzung der COM-Schnittstelle von Word Siehe Befehle der COM-Schnittstelle

In diesem Abschnitt wird Anhand von kleinen Beispielen die Benutzung der COM-Schnittstelle von Microsoft Word vorgestellt. Dabei wird nur ein kleiner Ausschnitt der Möglichkeiten vorgestellt.

Eine vollständige Beschreibung der von Microsoft Word zur Verfügung gestellten Objekte, deren Eigenschaften und Methoden, befindet sich im Objektkatalog des Visual Basic-Editors (Menüpunkt "Extras / Makro / Visual Basic-Editor" und anschließend Menüpunkt "Ansicht / Objektkatalog").

Herstellen einer Verbindung zu Microsoft Word

Die Verbindung zur COM-Schnittstelle von Microsoft Word wird über den Befehl ComOpen() hergestellt. Als Parameter wird der Applikationsname und das Root-Objekt angegeben. Mit der Option ComAppCreate wird ein neuer Word-Prozess gestartet.

```
tComApplication # ComOpen('Word.Application', _ComAppCreate);
```

Der zurückgegebene Deskriptor wird benötigt, um auf untergeordnete Objekte, Eigenschaften und Methoden des COM-Objektes "Application" zugreifen zu können.

Setzen einer Eigenschaft eines Objektes

Der neu gestartete Prozess wird nicht in der Taskleiste des Betriebssystems angezeigt. Damit die weitere Verarbeitung nachvollzogen werden kann, wird die Eigenschaft "Visible" des Objektes "Application" gesetzt:

```
tComApplication->cplVisible # true;
```

Eine Eigenschaft eines Objektes wird durch die Angabe des Objekt-Deskriptors, der Eigenschaft und des neuen Wertes gesetzt. In dem Beispiel wurde eine Konstante verwendet. Es kann aber auch der Befehl ComPropSet() verwendet werden.

Aufrufen einer Methode

Eine Methode wird mit dem Befehl ComCall() aufgerufen. Um ein neues Dokument zu erzeugen, muss die Methode "Add" des Objektes "Documents" aufgerufen werden. Das Objekt "Documents" ist dem Objekt "Application" untergeordnet. Die Methode kann auf unterschiedlichen Wegen aufgerufen werden. Ist der Objektpfad ausgehend vom Application-Objekt bekannt, kann dieser in der ComCall()-Anweisung angegeben werden:

```
tComApplication->ComCall('Documents.Add');
```

Die Objekte und die Methode werden durch einen Punkt voneinander getrennt. Wird der Deskriptor zum COM-Objekt "Documents" noch zu einem späteren Zeitpunkt benötigt, kann zunächst das entsprechende Objekt ermittelt und anschließend die Methode aufgerufen werden:

Kontakt

```
tComDocuments # tComApplication->cphDocuments;tComDocuments->ComCall('Add');
```

Abfragen von Ergebnissen einer Methode

Liefert eine aufgerufene Methode ein Ergebnis zurück, kann das Ergebnis mit der Anweisung ComCallResult() ermittelt werden.

```
tComApp # ComOpen('Word.Application', _ComAppCreate);tComApp->cplVisible # true; tResult # tComAp
```

Übertragen von formatierten Daten

Um Text in ein erzeugtes Dokument zu schreiben, muss die Methode "TypeText" des Objektes "Selection" aufgerufen werden. Vor dem Aufruf kann das Format des Textes geändert werden. Zunächst wird das Objekt "Selection" ermittelt. Aus dem Objekt "Selection" wird das Objekt "Font" bestimmt. Über die Eigenschaften des Font-Objektes kann das Format des Textes bestimmt werden.

```
tComSelection # tComApplication->cphSelection;tComFont # tComSelection->cphFont;tComFont->cpaName
```

Jetzt kann mit der Methode "TypeText" der Text übertragen werden:

```
tComSelection->ComCall('TypeText', 'Dear Sir or Madam,' + StrChar(13));
```

Speichern eines Dokumentes

Die Vorgehensweise zum Speichern eines Dokumentes entspricht der zum Übertragen von Text oder jedem anderen Aufruf einer Methode. Es wird das Objekt ermittelt, das verarbeitet werden soll und dessen Methode wird aufgerufen. Zum Speichern eines Dokumentes muss das Dokument ermittelt und die Methode "Save" oder "SaveAs" aufgerufen werden.

In diesem Beispiel soll das aktive Dokument gespeichert werden. Das Objekt kann über die Eigenschaft "ActiveDocument" des Objektes "Application" ermittelt werden. Der Aufruf zum Speichern des aktiven Dokumentes hat folgendes Aussehen.

```
tComApplication->ComCall('ActiveDocument.SaveAs', _Sys->spPathMyDocuments + '\Com_Example.doc');
```

Beenden der Verbindung

Wird die Verbindung zwischen CONZEPT 16 und Microsoft Word getrennt, wird Word nicht automatisch beendet. Es muss die Methode "Quit" des Objektes "Application" aufgerufen werden bevor die Verbindung mit ComClose() beendet wird.

```
tComApplication->ComCall('Quit', wdDoNotSaveChanged);tComApplication->ComClose();
```



Das Erstellen von Serienbriefen mit Hilfe der COM-Schnittstelle ist prinzipiell möglich. Durch die Art der Programmierung ist allerdings der Zugriff über die ODBC-Schnittstelle wesentlich flexibler, einfacher zu realisieren und damit auch einfacher zu pflegen. Durch entsprechende Abfragewerkzeuge, die auch von einem ungeübten Benutzer verwendet werden können, kann das Abfragen der Datenbank und das Erstellen von Serienbriefen, Katalogen usw., vollständig durch den Benutzer erfolgen.

Kontakt

Konstanten für COM-Befehle

Konstanten für die COM-Befehle

Siehe [COM-Befehle](#)

- [ComAppCreate](#)
- [ComInfoErrCode](#)
- [ComInfoErrText](#)

Kontakt

_ComAppCreate

Befehle zur Kommunikation mit der COM-Schnittstelle
Wert 1 /

0x00000001

Siehe [ComOpen\(\)](#)

Option bei [ComOpen\(\)](#) - Wird diese Option beim Befehl [ComOpen\(\)](#) angegeben, wird die angegebene Applikation gestartet.

Kontakt

_ComInfoErrCode

Fehlerwert ermitteln

Wert 0

Siehe [ComInfo\(\)](#)

Option bei [ComInfo\(\)](#) - Mit dieser Option wird der Fehlerwert ermittelt, der von der über COM angesteuerten Applikation erzeugt wurde. Die Meldung im Klartext kann über die Option [ComInfoErrText](#) ermittelt werden.

Kontakt

_ComInfoErrText

Fehlertext ermitteln

Wert 1

Siehe ComInfo()

Option bei ComInfo() - Mit dieser Option wird der von der über COM angesteuerten Applikation erzeugte Fehlertext ermittelt.

Kontakt

DDE-Befehle

Befehle um eine DDE-Ausgabe durchzuführen

Verwandte
Befehle, Liste
sortiert nach
Siehe Gruppen,
Alphabetische
Liste aller
Befehle

Befehle

- [DdeCommand](#)
- [DdeConnect](#)
- [DdeDisconnect](#)
- [DdeGetData](#)
- [DdeGetDataInfo](#)
- [DdeInit](#)
- [DdeServiceClose](#)
- [DdeServiceData](#)
- [DdeServiceOpen](#)
- [DdeServiceRead](#)
- [DdeSetData](#)
- [DdeTerm](#)

Konstanten

- [DdeAdviseOff](#)
- [DdeAdviseOn](#)
- [DdeAppend](#)
- [DdeCRLF](#)
- [DdeExecute](#)
- [DdeReceive](#)
- [DdeSend](#)
- [DdeServiceApp](#)
- [DdeServiceNext](#)
- [DdeServicePos](#)
- [DdeServicePrev](#)
- [DdeServiceTheme](#)

DDE ist die Abkürzung für "Dynamic Data Exchange" = "Dynamischer Datenaustausch".

CONZEPT 16 stellt Prozedurbefehle zur Verfügung, um mit einem DDE-Server kommunizieren zu können.

Kontakt

obj ->

DdeCommand(int1,



int2[, alpha3]) : int

DDE-Befehl absetzen

Deskriptor des

obj

DDE-Kanals

Funktionstyp

DdeReceive

Themeninhalt

abrufen

DdeSend

Information

übertragen

int1

DdeExecute

Kommando

ausführen

DdeAdviseOn

Aktualisierung

einschalten

DdeAdviseOff

Aktualisierung

ausschalten

int2

Maximale Wartezeit

Elementname bzw.

alpha3

optionales Kommando

Resultat int

Fehlerwert



Verwandte Befehle,

Siehe

DdeConnect()

Mit dieser Funktion wird ein DDE-Befehl an die DDE-Applikation geschickt. In (obj) steht der Deskriptor des DDE-Kanals. In (int1) wird der Funktionstyp übergeben:

- DdeReceive - Themeninhalt abrufen

Bei diesem Funktionstyp wird in (alpha3) der Elementname des mit DdeConnect() gewählten Themas angegeben. Die Information kann nach erfolgreicher Ausführung mit DdeGetData() aus dem Empfangs-Puffer ausgelesen werden.

- DdeSend - Information übertragen

Mit diesem Funktionstyp wird die Information aus dem Sendepuffer an die Applikation übertragen. In (alpha3) steht in diesem Fall das Element, in welches eingefügt werden soll (dabei kann es sich zum Beispiel um eine Textmarkierung handeln).

- DdeExecute - Kommando ausführen

Mit diesem Funktionstyp wird das Kommando in (alpha3) ausgeführt. Wird (alpha3) nicht angegeben, so wird das im Sendepuffer enthaltene Kommando ausgeführt (zum Beispiel lange Makroanweisungen).

- DdeAdviseOn - Aktualisierungsnachrichten einschalten

Mit dieser Funktion wird der DDE-Server beauftragt, bei jeder Datenänderung des Elements (alpha3) automatisch eine Nachricht zu versenden. Ist der DDE-Kanal für ein Fenster-Objekt geöffnet worden, wird beim Eintreffen einer Nachricht das Ereignis EvtAdviseDDE des Fenster-Objektes aufgerufen.

Kontakt

Ist der DDE-Kanal an die textorientierte Oberfläche von CONZEPT 16 gebunden, gibt der Befehl den Wert einer Funktionstaste zurück, die beim Eintreffen einer Aktualisierungsnachricht generiert wird (Damit der Funktionstastenwert auch verarbeitet werden kann, muss er gegebenenfalls per 'SetIoFKey' aktiviert werden).



Die Funktionstaste kann nur in der textbasierten Oberfläche ausgewertet werden.

- **DdeAdviseOff** - Aktualisierungsnachrichten ausschalten

Mit dieser Funktion wird der DDE-Server beauftragt, bei Datenänderungen des Elements (alpha3) keine Nachricht mehr zu versenden.

Im Argument (int2) kann eine maximale Wartezeit in Millisekunden angegeben werden. Wird die Wartezeit zu klein gewählt, kann als Resultat der Funktion -1 zurückgeliefert werden, obwohl der Befehl erfolgreich ausgeführt wurde. Bei erfolgreicher Ausführung des Befehls wird als Ergebnis Null zurückgeliefert.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj -> DdeConnect(alpha1, alpha2) : logic |  Verbindung

zu einer DDE-Applikation aufbauen

obj Deskriptor des DDE-Kanals

alpha1 DDE-Applikation

alpha2 DDE-Thema

Resultat logic Erfolg des Verbindungsbaus
Verwandte Befehle, DdeInit(),

Siehe

DdeDisconnect()

Mit diesem Befehl wird die Verbindung mit der in (alpha1) anzugebenden Applikation und Thema (alpha2) hergestellt. Der DDE-Kanal (obj) wird vorher mit DdeInit() geöffnet.

Beispiel:

Herstellen einer Verbindung zu dem in Word geöffneten Dokument

"Dokument1.DOC":

```
if (DdeConnect(tDdeHdl, 'WinWord', 'Dokument1')) { ... }
```

Zur Ermittlung von Applikation und Thema sind die Funktionen DdeServiceOpen(),
DdeServiceRead(), DdeServiceData() und DdeServiceClose() vorhanden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj -> DdeDisconnect()

Verbindung abbauen

Verbindung abbaubar Deskriptor des obj

DDE-Kanals

Verwandte

Siehe Befehle.

Berichte,

[DacConnect](#) Mit diesem Befehl wird die Verbindung für den DDE-Kanal (obj) abgebaut. Für diesen Kanal muss vorher eine Verbindung aufgebaut worden sein, ansonsten erfolgt ein Laufzeitfehler.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj -> DdeGetData(int1, int2, int3) :



alpha

Inhalt des DDE-Empfangspuffers lesen

obj Deskriptor des DDE-Kanals

int1 Zeilennummer

int2 Startposition innerhalb der Zeile

int3 Anzahl der zu lesenden Zeichen

Resultat alpha Inhalt des Empfangspuffers

Siehe [Verwandte Befehle](#), [DdeSetData\(\)](#)

Mit dieser Funktion lassen sich die im Empfangs-Puffer enthaltenen Informationen auslesen. In (int1) wird die Zeile, in (int2) die Position des ersten zu lesenden Zeichens in der Zeile angegeben. In (int3) steht die Anzahl der zu lesenden Zeichen. Die Länge einer Zeile im Empfangspuffer ist nicht auf 250 Stellen begrenzt.

Mögliche Laufzeitfehler:

[ErrHdlInvalid](#) Deskriptor ungültig

Kontakt

obj -> DdeGetDataInfo([int1]) : int

Informationsumfang von DDE-Daten ermitteln

obj Deskriptor des DDE-Kanals

int1 Zeilennummer (optional)

Resultat int Zeilenanzahl bzw. Zeilenlänge

Siehe [Verwandte Befehle](#)

Mit diesem Befehl wird die Anzahl der Zeilen im Empfangspuffer ermittelt. Sofern als (int1) eine Zeilennummer übergeben wird, entspricht das Resultat der Länge dieser Zeile.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj -> DdeInit(int1) : handle 

Schnittstelle initialisieren

obj Objekt

int1 reserviert (0)

Resultat handle Deskriptor des neuen

DDE-Kanals

Siehe Verwandte Befehle, DdeTerm()

Mit diesem Befehl wird eine DDE-Schnittstelle initialisiert, indem ein DDE-Kanal geöffnet wird, dessen Deskriptor bei allen weiteren DDE-Operationen benötigt wird. Das Resultat ist -1, wenn die Initialisierung fehlschlägt.

Wird der Befehl aus einem Dialog heraus verwendet, muss als erster Parameter der Deskriptor des Dialoges übergeben werden. Der entsprechende Dialog muss zu diesem Zeitpunkt sichtbar sein. Die Ausführung des Befehls kann also nicht in dem Ereignis EvtInit des gleichen Fensters erfolgen.

Der Befehl darf nicht verwendet werden, wenn noch Ereignisse des Betriebssystems ausgeführt werden. Dies ist zum Beispiel unmittelbar nach dem Schließen eines Fensters der Fall. Soll trotzdem eine DDE-Verbindung aufgebaut werden, muss nach dem Schließen des Fensters mit dem Befehl SysSleep() das Abarbeiten der Ereignisse abgewartet werden.

Beispiel:

```
DdeInit(tFrame, 0) // oder tFrame->DdeInit(0)
```

Wird der Befehl aus Masken heraus aufgerufen muss die Anweisung wie folgt lauten:

```
DdeInit(0, 0)
```

Kontakt

obj -> DdeServiceClose() |  |  |  | DDE- |  |  |  | 

Servicefunktionen beenden

obj Deskriptor des

DDE-Kanals

Siehe [Verwandte Befehle](#),
[DdeServiceOpen\(\)](#)

Mit diesem Befehl werden die Servicefunktionen für den DDE-Kanal (obj) beendet. Vor Verlassen von CONZEPT 16 müssen alle bestehenden Servicefunktionen beendet werden, da sonst alle temporär in der Datenbank gespeicherten Serviceinformationen zu diesem DDE-Kanal in der Datenbank verbleiben.



Ein späteres Entfernen von temporären Service-Daten ist nur durch eine Datenbankoptimierung möglich.

Mögliche Laufzeitfehler:

[ErrHdlInvalid](#) Deskriptor ungültig

Kontakt

obj -> DdeServiceData(int1) :



alpha

DDE-Serviceinformation abfragen

obj Deskriptor des DDE-Kanals
 DdeServiceApp Applikation

int1 DdeServiceTheme Thema

Resultat alpha Serviceinformation

Verwandte Befehle,

Siehe

DdeServiceRead()

Mit dieser Funktion kann die Applikation oder das Thema des gelesenen Service-Eintrages (siehe DdeServiceRead()) ermittelt werden. In (int1) ist der Typ der gewünschten Information anzugeben.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj ->

DdeServiceOpen([alpha1[,
alpha2]]) : logic



DDE-Servicefunktionen starten

obj Deskriptor des DDE-Kanals
alpha1 DDE-Applikation (optional)
alpha2 DDE-Thema (optional)

Resultat logic Erfolg des Kommandos
Verwandte Befehle,

Siehe

DdeServiceClose()

Mit diesem Befehl werden die DDE-Servicefunktionen initialisiert. Mit den Servicefunktionen können Informationen über die aktuell verfügbaren DDE-Server (Applikation) sowie deren Themen (Topic) abgefragt werden.

Ein Thema kann z. B. der Name eines Dokuments in einer Textverarbeitung sein. Das Thema "System" ist in allen DDE-Applikationen enthalten. In (obj) ist der Deskriptor des DDE-Kanals anzugeben. In (alpha1) kann der Name der Applikation und in (alpha2) der Name des Themas übergeben werden, auf den der Service beschränkt werden soll. Wird nur (obj) übergeben, so können mit den Service-Funktionen alle aktuellen DDE-Applikationen angesprochen werden.

Beispiel:

```
if (DdeServiceOpen(tKanal, 'WinWord', '')){ ... }
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj -> DdeServiceRead(int1[,

alpha2[, alpha3]] : int DDE-

Serviceinformation lesen

obj Deskriptor des DDE-Kanals

Lesemodus

DdeServicePos direkt

lesen

int1 DdeServiceNext nächster

Eintrag

DdeServicePrev vorheriger

Eintrag

alpha2 Applikationsname (optional)

alpha3 Themenname (optional)

Resultat int Fehlerwert

Verwandte Befehle,

Siehe DdeServiceOpen(),

DdeServiceData()

Mit dieser Funktion wird ein Eintrag des mit DdeServiceOpen() geöffneten

Verzeichnisses gelesen. In (int1) steht der zu verwendende Lesemodus:

- DdeServicePos - Eintrag direkt lesen

Hierbei wird direkt auf den in (alpha3) bzw. (alpha3) übergebenen Eintrag positioniert.

Sofern weder (alpha2) noch (alpha3) angegeben werden, wird der allererste Eintrag gelesen.

- DdeServiceNext - nächsten Eintrag lesen

Hierbei wird der nachfolgende Eintrag gelesen; die Argumente (alpha2) und (alpha3) werden dabei weggelassen.

- DdeServicePrev - vorherigen Eintrag lesen

Hierbei wird der vorherige Eintrag gelesen; die Argumente (alpha2) und (alpha3) werden dabei weggelassen.

Das Resultat ist 0, wenn die Operation erfolgreich war.

Die Werte des gelesenen Eintrags können anschließend mit DdeServiceData() abgefragt werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig



Kontakt

obj -> DdeSetData(alpha1, int2) |  Daten in den _____

DDE-Sendepuffer übertragen

Deskriptor des
obj

DDE-Kanals

alpha1 Daten

Optionen

DdeAppend Daten

int2 anhängen

DdeCRLF CR + LF
anhängen

Siehe [Verwandte Befehle](#),

[DdeGetData\(\)](#)

Mit dieser Funktion werden die Daten (alpha1) in den Sendepuffer übertragen. Durch Angabe der Option DdeAppend in (int2) wird der Wert in (alpha1) zu den bereits im Puffer befindlichen Daten hinzugefügt, andernfalls wird der Pufferinhalt geleert. Mit der Option DdeCRLF wird ein zusätzliches Zeilenende (CR + LF) angehängt. Der Sendepuffer vergrößert sich automatisch nach Bedarf.

Mögliche Laufzeitfehler:

[ErrHdlInvalid Deskriptor ungültig](#)

Kontakt

obj -> DdeTerm()

DDE-Schnittstelle beenden

Deskriptor

obj des

DDE-Kanals

Verwandte

Siehe Befehle,

DdeInit()

Mit diesem Befehl wird ein mittels DdeInit() geöffneter DDE-Kanal wieder geschlossen. Ist beim Öffnen des DDE-Kanals ein Fenster angegeben worden, muss dieses Fenster zum Zeitpunkt der Ausführung dieses Befehls sichtbar sein. Der Befehl kann also nicht in dem Ereignis EvtTerm des gleichen Fensters erfolgen. Statt dessen kann das Ereignis EvtClose verwendet werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

Konstanten für DDE-Befehle

Siehe <u>Alle Befehle</u>	
<u>DdeAdviseOff</u>	Aktualisierungsnachrichten ausschalten
<u>DdeAdviseOn</u>	Aktualisierungsnachrichten einschalten
<u>DdeAppend</u>	Daten anhängen
<u>DdeCrLf</u>	CR + LF anhängen
<u>DdeExecute</u>	Kommando ausführen
<u>DdeReceive</u>	Themeninhalt abrufen
<u>DdeSend</u>	Informationen übertragen
<u>DdeServiceApp</u>	Applikation ermitteln
<u>DdeServicePos</u>	Eintrag direkt lesen
<u>DdeServiceNext</u>	nächsten Eintrag lesen
<u>DdeServicePrev</u>	vorherigen Eintrag lesen
<u>DdeServiceTheme</u>	Thema ermitteln

Kontakt

_DdeAdviseOff

Aktualisierungsnachricht

Wert 5

Verwandte

Siehe Befehle,

DdeCommand()

Option bei DdeCommand() - Aktualisierungsnachrichten ausschalten.

Kontakt

_DdeAdviseOn

Aktualisierungsnachricht

Wert 4

Verwandte

Siehe Befehle,

DdeCommand()

Option bei DdeCommand() - Aktualisierungsnachrichten aktivieren. Dieses Kommando wird verwendet, um einem DDE-Server die Möglichkeit zu geben, die Applikation zu informieren, wenn Daten zur Verfügung gestellt wurden.

Der DDE-Server stellt dazu entsprechende Kommunikationselemente zur Verfügung.

Dieses Element muss bei dem DdeCommand()-Befehl angegeben werden.

Stehen Daten zur Verfügung, wird das Ereignis EvtAdviseDDE des Fenster-Objektes, für den der DDE-Kanal geöffnet wurde (DdeInit()), aufgerufen.

Beispiel:

```
DdeCommand(tDdeHdl, _DdeAdviseOn, 500, 'DdeServerItem');
```

Kontakt

[_DdeAppend](#)

Daten anhängen

Wert 1

[Verwandte](#)

Siehe [Befehle](#),

[DdeSetData\(\)](#)

Option bei [DdeSetData\(\)](#). Die übertragenen Daten werden im Puffer an vorhandene Daten angehängt.

Kontakt

_DdeCRLF

Übertragungsformat

Wert 2

Verwandte

Siehe Befehle,

DdeSetData()

Option bei DdeSetData() - Daten mit CR + LF anhängen

Kontakt

_DdeExecute

Kommando ausführen

Wert 3

Verwandte

Siehe Befehle,

DdeCommand()

Option von DdeCommand(). Mit dieser Option wird das Kommando, das im Befehl DdeCommand() angegeben ist, ausgeführt.

Kontakt

_DdeReceive

Themeninhalt

Wert 1

Verwandte

Siehe Befehle,

DdeCommand()

Option bei DdeCommand() - Themeninhalt abrufen

Kontakt

_DdeSend

Informationen übertragen

Wert 2

Verwandte

Siehe Befehle,

DdeCommand()

Option bei DdeCommand(). Mit dieser Option werden die Informationen, die zuvor in den Puffer geschrieben wurden, übertragen.

Kontakt

_DdeServiceApp

Applikation

Wert 1

Verwandte

Siehe **Befehle**,

DdeServiceData()

Option bei **DdeServiceData()** - Applikation ermitteln

Kontakt

_DdeServiceNext

Eintrag lesen

Wert 1

Siehe [Verwandte Befehle](#),

Option bei [DdeServiceRead\(\)](#) - nächsten Eintrag lesen

Kontakt

_DdeServicePos

Eintrag lesen

Wert 0

Siehe Verwandte Befehle,

Option bei DdeServiceRead() - Eintrag direkt lesen

Kontakt

_DdeServicePrev

Eintrag lesen

Wert 2

Siehe [Verwandte Befehle](#),

Option bei [DdeServiceRead\(\)](#) - vorherigen Eintrag lesen

Kontakt

_DdeServiceTheme

Thema

Wert 2

Verwandte

Siehe Befehle,

DdeServiceData()

Option bei DdeServiceData() - Thema ermitteln

DLL-Befehle

Befehle zum Aufruf einer DLL

[Verwandte](#)

[Befehle](#)

[Beispiel, Liste](#)

Siehe [sortiert nach](#)

[Gruppen](#),

[Alphabetische](#)

[Liste aller](#)

[Befehle](#)

Mit den hier aufgeführten Befehlen kann eine externe DLL in CONZEPT 16 integriert werden. Als Schnittstelle zwischen der CONZEPT 16-Prozedur und den Funktionen der DLL muss in der DLL eine Funktion geschrieben werden, die von außerhalb der DLL aufgerufen werden kann. Weitere Informationen befinden sich im Abschnitt [DLL in CONZEPT 16 einbinden](#).

Befehle

- [DllCall](#)
- [DllLoad](#)
- [DllUnload](#)

Kontakt

DLL in CONZEPT 16 einbinden

Aufbau einer DLL, die in CONZEPT 16 eingebunden werden kann.

Verwandte

Siehe [Befehle](#),

[Beispiel](#)

Mit den [DLL-Befehlen](#) kann in CONZEPT 16 eine DLL eingebunden und Funktionen aus dieser DLL aufgerufen werden. Die DLL muss dazu bestimmte Bedingungen erfüllen.

In der DLL muss eine Funktion aufgerufen werden können, die als Parameter den "Call Control Block" übergeben bekommt. Der Funktionsname muss beim Laden der DLL mit [DllLoad\(\)](#) angegeben werden. Wird als Funktionsname C16_DLLCALL (Groß-/Kleinschreibung ist zu beachten) verwendet, kann die Angabe des Namens entfallen. Über diese Einstiegsfunktion werden alle Funktionen der DLL aufgerufen. Der "Call Control Block" und weitere Definitionen befinden sich in der Datei [c16_dll.h](#), die in die DLL eingebunden werden muss.

Der Befehl [DllLoad\(\)](#) gibt den Instanz-Handle der DLL zurück. Dieser muss beim Befehl [DllCall\(\)](#) angegeben werden. Der Befehl ruft nur die Einstiegsfunktion auf.

Beispiel:

Aufruf in CONZEPT 16

```
tD11 # DllLoad('C:\c16\57\mydll');if (tD11 <= 0){ // Fehler beim Laden der DLL ...}else{ DllC
```

In der Header-Datei [c16_dll.h](#) befinden sich alle notwendigen Definitionen. Eine zentrale Bedeutung hat hier der "Call Control Block", der der Einstiegsfunktion übergeben wird. In dieser Struktur befinden sich unter anderem alle Sprungadressen, um verschiedene Funktionen des CONZEPT 16-Clients aufrufen zu können.



Beim Compilieren der DLL ist darauf zu achten, dass die Einstiegsfunktion von außerhalb der DLL aufgerufen werden kann. Dies geschieht in der Regel durch die Angabe von Optionen beim Linken.

Der Call Control Block

Der Call Control Block enthält die Instanz der geladenen DLL (InstHdl). Die Instanz muss bei allen Funktionsaufrufen in den CONZEPT 16-Client angegeben werden. Des Weiteren sind eine Reihe von Funktionszeigern enthalten. Die Funktionen entsprechen in ihrer Wirkungsweise den gleichnamigen Funktionen der [Externen Programmierschnittstelle](#).

Zum Aufruf der entsprechenden Funktionen muss vom Call-Control-Block ausgegangen werden. Im Fall von C und C++ wird die Funktion zum Lesen eines Datensatzes mit aCCB->C16_RecRead(...); aufgerufen.

Funktionen aus folgenden Bereichen können aufgerufen werden:

- [Funktionen zur Verarbeitung von Übergabeparameter](#)
- [Datensatz-Funktionen](#)
- [Satzpuffer-Funktionen](#)
- [Funktionen zum Ermitteln von Informationen](#)

Kontakt

- Text-Funktionen
- Transaktions-Funktionen
- Sonstige Funktionen

Funktionen zur Verarbeitung von Übergabeparametern

C16_ArgCount() Anzahl der übergebenen Argumente ermitteln

C16_ArgInfo() Informationen zu einem Argument ermitteln

C16_ArgRead() Argument lesen

C16_ArgWrite() Argument schreiben

Datensatz-Funktionen

C16_ExcFileData() Veranlasst den Transfer von Daten zwischen allen registrierten Programmvariablen und den jeweiligen Feldern im Feldpuffer

C16_ExcFldData() Veranlasst den Transfer von Daten zwischen einer registrierten Programmvariable und dem mit ihr verbundenen Feld im Feldpuffer

C16_ExcSbrData() Veranlasst den Transfer von Daten zwischen allen registrierten Programmvariablen und den jeweiligen Feldern eines Teildatensatzes

C16_FldData() Feldinhalte lesen und schreiben (binär)

C16_FldDataChar() Feldinhalte lesen und schreiben mit ASCII-Konvertierung

C16_FldDataRaw() Speicherung von Binärdaten in alphanumerischen Feldern

C16_RecDelete() Datensatz löschen

C16_RecDeleteAll() Alle Datensätze einer Datei löschen

C16_RecInsert() Datensatz einfügen

C16_RecLink() Verknüpften Datensatz lesen

C16_RecRead() Datensatz lese

C16_RecReplace() Datensatz zurückspeichern

C16_SbrSetStatus() Status eines Teildatensatzes setzen

C16_SetExcMode() Automatischer Transfermodus aktivieren / deaktivieren

Satzpuffer-Funktionen

C16_RecBufClear() Satzpuffer leeren

C16_RecBufCompare() Pufferinhalte vergleichen

C16_RecBufCopy() Pufferinhalte kopieren

C16_RecBufCreate() Satzpuffer anlegen

C16_RecBufDestroy() Satzpuffer entfernen

C16_RegExtFld() Erstellt eine direkte Verbindung zwischen einer im Programm definierten Variable und einem Feld im Feldpuffer

C16_SbrClear() Teildatensatzpuffer leeren

Funktionen zum Ermitteln von Informationen

C16_FileInfo() Ermittelt Informationen zu einer Datei

C16_FileInfoByName() Ermittelt Informationen zu einer Datei über den Namen

Kontakt

<u>C16_FldInfo()</u>	Ermittelt Informationen zu einem Feld
<u>C16_FldInfoByName()</u>	Ermittelt Informationen zu einem Feld über den Namen
<u>C16_KeyFldInfo()</u>	Ermittelt Informationen zu einem Schlüsselfeld
<u>C16_KeyInfo()</u>	Ermittelt Informationen zu einem Schlüssel
<u>C16_KeyInfoByName()</u>	Ermittelt Informationen zu einem Schlüssel über den Namen
<u>C16_LinkFldInfo()</u>	Ermittelt Informationen zu einem Verknüpfungsfeld
<u>C16_LinkInfo()</u>	Ermittelt Informationen zu einer Verknüpfung
<u>C16_LinkInfoByName()</u>	Ermittelt Informationen zu einer Verknüpfung über den Namen
<u>C16_QueryAreaInfo()</u>	Ermittelt Informationen zu einer Datenbank
<u>C16_QueryPgInfo()</u>	Ermittelt Informationen zu der Programmierschnittstelle
<u>C16_QueryServerInfo()</u>	Ermittelt Informationen zum Server
<u>C16_QueryUserInfo()</u>	Ermittelt Informationen zu Benutzern
<u>C16_RecInfo()</u>	Ermittelt Informationen zu einem Datensatz
<u>C16_RecLinkInfo()</u>	Ermittelt Informationen zu einem verknüpften Datensatz
<u>C16_SbrInfo()</u>	Ermittelt Informationen zu einem Teildatensatz
<u>C16_SbrInfoByName()</u>	Ermittelt Informationen zu einem Teildatensatz über den Namen

Text-Funktionen

<u>C16_TextClose()</u>	Textpuffer entfernen
<u>C16_TextCopy()</u>	Text kopieren
<u>C16_TextCreate()</u>	Leeren Text erzeugen
<u>C16_TextDelete()</u>	Text löschen
<u>C16_TextOpen()</u>	Textpuffer erzeugen
<u>C16_TextRead()</u>	Textinformationen lesen
<u>C16_TextReadData()</u>	Inhalt eines Textes lesen
<u>C16_TextRename()</u>	Text umbenennen
<u>C16_TextWrite()</u>	Textinformationen schreiben
<u>C16_TextWriteData()</u>	Inhalt eines Textes schreiben

Transaktions-Funktionen

<u>C16_DtaBegin()</u>	Transaktion starten
<u>C16_DtaCommit()</u>	Aktive Transaktion beenden
<u>C16_DtaRollback()</u>	Aktive Transaktion abbrechen

Sonstige Funktionen

<u>C16_ProcArgument()</u>	Prozedurargument übergeben
<u>C16_ProcCall()</u>	CONZEPT 16-Prozedur aufrufen
<u>C16_ProcResult()</u>	Prozedurergebnis ermitteln
<u>C16_ServerBackup()</u>	Sicherungsereignis des Servers starten
<u>C16_SetCharDefs()</u>	Zeichenformat für ASCII-Konvertierung festlegen
<u>C16_UserClear()</u>	Benutzer aus der Datenbank entfernen

Kontakt

c16_dll.h

Header-Datei zum Erstellen einer DLL

Die Header-Datei bindet eine weitere Header-Datei (c16.h) mit allgemeinen Definitionen ein.

```
*****/*  
vERROR (C16API_DLL * C16_ArgRead)( const vPHANDLE aInstHdl, /* in: instance  
/* retrieve subrecord information by name */vERROR (C16API_DLL * C16_SbrInfoByName)( const vPHANDLE  
*****/* KEY information  
/* retrieve link field information */vERROR (C16API_DLL * C16_LinkFldInfo)( const vPHANDLE  
/* read/write field data as chars */vERROR (C16API_DLL * C16_FldDataChar)( const vPHANDLE  
/* transfer subrecord */vERROR (C16API_DLL * C16_ExcSbrData)( const vPHANDLE aInstHdl,  
/* retrieve linked record information */vERROR (C16API_DLL * C16_RecLinkInfo)( const vPHANDLE  
/* update record */vERROR (C16API_DLL * C16_RecReplace)( const vPHANDLE aInstHdl,  
/* create temporary record buffer */vERROR (C16API_DLL * C16_RecBufCreate)( const vPHANDLE  
/* copy text */vERROR (C16API_DLL * C16_TextCopy)( const vPHANDLE aTextHdl,  
*****/* procedures  
/* query user informations */vERROR (C16API_DLL * C16_QueryUserInfo)( const vPHANDLE aIns  
/* retrieve record information */vERROR (C16API_DLL * C16_RecInfo64)( const vPHANDLE aIns
```

Kontakt

c16.h

Header-Datei von CONZEPT 16

```
*****/*  
#ifdef _CHAR_UNSIGNEDtypedef char vCHAR; /* 8 bit unsigned  
// Structure of _TypeDecimaltypedef struct{ vBYTE length; vBYTE sign; vBYTEs exponent; vBYT  
*****/*  
#define C16ERR_NO SERVER CONNECTION -401#define C16ERR AREA NOT FOUND -402#define  
#define C16ERR_BIN_NAME_INVALID -851#define C16ERR_BIN_NO_PATH -852#define  
// compiler#define C16ERR_PCDC_CODE_OVERFLOW -1106#define C16ERR_PCDC_CONST_VAL  
*****/* - P-CODE INFORMATION  
#define C16ERR_LICENSE_SRVNOTSUPPORTED -2001#define C16ERR_LICENSE_CLNUPGRADE -2002#define  
#define _FileAttrOemMark 0x01#define _FileAttrTemp 0x02#define  
#define _RecFirst 0x00000001#define _RecLast 0x0000  
#define _RecCount 0#define _RecCountPos 5#define _RecC  
#define _SelFirst 0x01#define _SelLast 0x02#define  
#define _PgifType_Client 0x0001 /* Client-type */#define _PgifyTy  
#define _DiagRes_End 0#define _DiagRes_TreeCollision 1#define _Diag  
/* field information structure */typedef struct{ vINT InfoSize;  
/* key field information structure */typedef struct{ vINT InfoSize;  
/* text header information structure */typedef struct{ vINT InfoSize;  
/* interface information structure */typedef struct{ vINT InfoSize;  
/* user information structure */typedef struct{ vINT InfoSize;  
/* diagnose progress information structure */typedef struct{ vINT InfoSize;  
/* compiler information structure */typedef struct{ vERROR ErrorCode; vINT
```

Kontakt

`obj -> DllCall([[, var]]) : int`



Funktion einer DLL oder einem SO aufrufen

obj Deskriptor der DLL

var Argumente (optional)

Resultat int Rückgabewert der DLL-Funktion

[Verwandte Befehle](#),

Siehe [DLL-Funktionen](#),

[Argument-Funktionen, Beispiel](#)

Mit diesem Befehl kann die Einstiegsfunktion einer Dynamic Link Library (Windows - DLL) oder einem Shared Object (Linux - SO) aufgerufen werden. Die Bibliothek muss zuvor mit dem Befehl [DllLoad\(\)](#) geladen worden sein. Der von diesem Befehl zurückgegebene Deskriptor wird in (obj) übergeben. Der Einstiegsfunktion der Bibliothek können bis zu 23 weitere Argumente übergeben werden. In der Bibliothek können die Anzahl der Argumente ([C16 ArgCount\(\)](#)) und Informationen über die Argumente ([C16 ArgInfo\(\)](#)), wie zum Beispiel der Typ des Parameters, ermittelt werden.

Die Argumente können mit den Funktionen [C16 ArgRead\(\)](#) gelesen und mit der Funktion [C16 ArgWrite\(\)](#) geschrieben werden. Um Argumente an die aufrufende Prozedur zurückzugeben, müssen diese Parameter mit [var](#) übergeben worden sein.

Bei umfangreicherer Bibliotheken sollte im ersten optionalen Argument ein Code zur Unterscheidung von verschiedenen Funktionen innerhalb der Bibliothek übergeben werden. Die Einstiegsfunktion überprüft dann das erste Argument und entscheidet in einem CASE-Konstrukt, welche Funktion innerhalb der Bibliothek aufgerufen werden soll.

Kontakt

DllLoad(alpha1[,
alpha2]) : handle DLL
oder SO laden



Pfad und Name der DLL

alpha1 (Windows) oder des SO
(Linux)
Name der

alpha2 Einstiegsfunktion

(optional)
Resultat handle Deskriptor oder

ErrGeneric

Verwandte Befehle,

Siehe

DllCall(), DllUnload()

Der Befehl lädt die in (alpha1) übergebene Dynamic Link Library (Windows - DLL) oder das Shared Object (Linux - SO) in den Speicher. Der Name der Bibliothek kann unter Windows ohne Dateierweiterung (.DLL) angegeben werden. In (alpha2) kann der Name der Einstiegsfunktion in der Bibliothek definiert werden. Wird keine Einstiegsfunktion angegeben, wird die Funktion C16_DLLCALL angesprungen.

Wird die Bibliothek vom SOA-Service mit dem Konfigurationseintrag c16_proc_library vorgeladen, wird in (alpha1) eine leere Zeichenkette übergeben.

Als Resultat wird der Deskriptor der Bibliothek zurückgegeben. Wurde die Bibliothek oder die Einstiegsfunktion nicht gefunden, wird ErrGeneric zurückgegeben.

Mit der Eigenschaft FsiError des Sys-Objektes kann der Fehler, der vom Betriebssystem liefert wird, abgefragt werden. Häufige Fehler sind:

- 126 - DLL nicht vorhanden, bzw. eine von der DLL referenzierte DLL ist nicht vorhanden.
- 127 - DLL-Einstiegsfunktion (alpha2) ist nicht vorhanden.
- 193 - DLL ist beschädigt oder Plattform stimmt nicht überein (32-/64-bit).

Nach dem Laden der Bibliothek kann mit dem Befehl DllCall() die Einstiegsfunktion aufgerufen werden.

Beispiel:

```
tDllHdl # DllLoad('c:\c16\dll\mydll', 'C16_DLLCALL');if (tDllHdl <= _ErrOk){ // Betriebssystem-F
```



Hat die geladene Bibliothek abhängige Bibliotheken, genügt es nicht wenn diese im gleichen Verzeichnis liegen. Die Abhängigkeiten müssen in einem Ordner liegen, der in der Umgebungsvariable Path (Windows) bzw. LD_LIBRARY_PATH (Linux) enthalten ist, oder mit FsiPathChange() als aktuelles Verzeichnis definiert wird.

Kontakt

obj -> DllUnload()

DLL oder SO entladen

Deskriptor
der DLL

obj (Windows)
oder des
SO (Linux)

Verwandte

Siehe **Befehle,**
DllLoad()

Mit diesem Befehl wird die in (obj) übergebene Dynamic Link Library (Windows - DLL) oder das Shared Object (Linux - SO) entladen. Der Deskriptor wird von dem Befehl **DllLoad()** zurückgegeben. Wird in (obj) ein falscher Deskriptor übergeben, erfolgt der Laufzeitfehler **ErrHdIInvalid**.

E-Mail-Befehle

Befehle zum Verschicken von E-Mails

Verwandte
Befehle, Liste
sortiert nach

Siehe Gruppen,

Alphabetische
Liste aller
Befehle

Der Versand von E-Mails wird über das TCP/IP-Protokoll SMTP (Simple Mail Transfer Protocol) abgewickelt.

Voraussetzung ist dass ein SMTP-fähiger Mailserver vorhanden und erreichbar ist.

Für den Versand von externen Mails (Empfänger außerhalb des lokalen Netzwerks) muss der Mailserver diese entsprechend weiterleiten können (z. B. in das Internet).

Nachrichten werden grundsätzlich im MIME-Format (Multi-purpose Internet Mail Extension) erzeugt, so dass der Empfänger über einen MIME-fähigen Mail-Client verfügen muss.

Befehle

- MailClose
- MailData
- MailOpen

Konstanten

- MailAuthNTLM
- MailCreateEML
- MailSsmtp
- MailSsmtpS
- MailSsmtpTls
- SckNoTLSv1
- SckTlsHigh
- SckTlsLow
- SckTlsMax
- SckTlsMed
- SckTlsSNI
- SmtpBCC
- SmtpCC
- SmtpDiscard
- SmtpFrom
- SmtpPriority
- SmtpReplyTo
- SmtpSendNow
- SmtpSubject
- SmtpTo

Kontakt

- [MailBuffer](#)
- [MailFile](#)
- [MailFileEML](#)
- [MailLine](#)
- [MimeTextHtml](#)
- [MimeApp](#)
- [MimeAppMSWORD](#)
- [MimeAppPDF](#)
- [MimeAppPS](#)
- [MimeAppRTF](#)
- [MimeAppZIP](#)
- [MimeContentID](#)
- [MimeCS IBM437](#)
- [MimeCS IBM850](#)
- [MimeCS ISO 8859 1](#)
- [MimeCS UTF8](#)
- [MimeImageGIF](#)
- [MimeImageJPEG](#)
- [MimeImagePNG](#)
- [MimeImageTIFF](#)
- [MimeOther8B](#)
- [MimeOtherB64](#)
- [MimeOtherQP](#)
- [MimeRelated](#)
- [MimeTE 8B](#)
- [MimeTE B64](#)
- [MimeTE OP](#)

Kontakt

**MailOpen(int1, alpha2[,
int3[, int4[, alpha5,
alpha6]]]) : handle**
Neue E-Mail erzeugen



	Mail- / Authentifizierungstyp
	<u>MailSmtp</u> Unverschlüsselte
	<u>MailSmtpTls</u> Verbindung aufbauen Verschlüsselte Verbindung mit STARTTLS aufbauen, wenn unterstützt
	<u>MailSmtpS</u> Verschlüsselte Verbindung mit SMTPS aufbauen
	<u>MailAuthNTLM</u> Authentifizierung mit NTLM verwenden
	<u>MailCreateEML</u> E-Mail als EML-Datei
	<u>SekTlsMax</u> speichern Verschlüsselung mit maximalen
int1	<u>SekTlsHigh</u> Sicherheitsanforderungen Verschlüsselung mit hohen
	<u>SekTlsMed</u> Sicherheitsanforderungen Verschlüsselung mit durchschnittlichen
	<u>SekTlsLow</u> Sicherheitsanforderungen Verschlüsselung mit niedrigen
	<u>SekNoTLSv1</u> Sicherheitsanforderungen Verschlüsselung ohne
	<u>SekTlsSNI</u> Unterstützung von TLS 1.x Verwendung von Server Name Indication (SNI) beim verschlüsselten Verbindungsaufbau
	Name oder IP-Adresse des
alpha2	Mailservers / Name der EML-Datei
int3	SMTP-Port (optional)
int4	Timeout in Sekunden (optional)
alpha5	Benutzername (optional)
alpha6	Kennwort (optional)
Resultat handle	Mail-Deskriptor <u>Verwandte Befehle, MailClose()</u> ,

Siehe **MailData()**

Um eine Mail zu erstellen und zu versenden, wird zunächst ein neuer Mail-Puffer erzeugt. Die maximale Größe des Mail-Puffers beträgt 32 MB (unter Linux-Systemen 8 MB). Es können somit keine E-Mails versendet werden, die einschließlich aller Anhänge größer sind als 32 MB. Das Limit von 8 MB greift, wenn eine Funktion mit RmtCall() auf dem Server ausgeführt wird, die eine E-Mail versendet.

Kontakt

Als Mail-Typ (int1) können folgende Konstanten angegeben werden:

<u>MailSmtp</u> (0)	Unverschlüsselte Verbindung aufbauen (Standardport 25, falls nicht angegeben)
<u>MailSmtpTls</u> (1)	Verschlüsselte Verbindung mit STARTTLS aufbauen, wenn unterstützt (Standardport 25, falls nicht angegeben)
<u>MailSmtpS</u> (2)	Verschlüsselte Verbindung mit SMPTS aufbauen (Standardport 587, falls nicht angegeben)
<u>MailCreateEML</u> (3)	E-Mail als EML-Datei speichern statt sie zu versenden

Wenn der Mailserver verschlüsselte Verbindungen unterstützt, wird mit MailSmtpTls eine verschlüsselte Verbindung aufgebaut, ansonsten eine unverschlüsselte Verbindung. Zu beachten ist, dass der Server mindestens SSL 3.0 oder TLS 1.x unterstützt und ein gültiges Zertifikat besitzt. Es findet eine Prüfung des Gültigkeitszeitraums statt, jedoch keine Überprüfung des Ausstellers oder des Domainnamens. Somit sind auch selbst ausgestellte Zertifikate möglich. Akzeptiert ein Mailserver nur verschlüsselte Verbindungen, muss diese Konstante angegeben werden. Geschieht dies nicht, kommt beim Befehl MailClose() der Fehler 58.



Durch das Handshake bei STARTTLS dauert der Versand der E-Mail etwas länger. Daher wird MailSmtpTls nur benötigt, wenn der Mailserver eine verschlüsselte Verbindung voraussetzt, oder sensible Inhalte zu einem externen Mailserver übertragen werden.

Wird MailSmtpS angegeben, wird die Nachricht per SMPTS-Verfahren versendet. Dabei wird beim Verbindungsauftbau eine Verschlüsselung per SSL / TLS verwendet. Hier gelten die gleichen Einschränkungen bezüglich SSL / TLS und dem verwendeten Zertifikat wie bei MailSmtpTls. Die Kommunikation findet üblicherweise auf dem 587 statt. Ältere Server verwenden unter Umständen den Port 465.

Die Optionen MailSmtpTls und MailSmtpS können mit einer der Sicherheitsstufen SckTlsMax, SckTlsHigh, SckTlsMed oder SckTlsLow kombiniert werden. Ist keine der Optionen angegeben, wird SckTlsMed verwendet. Zusätzlich kann zu SckTlsMed und SckTlsLow SckNoTLSv1 angegeben werden, falls der Server Probleme mit den TLS 1.x-Protokollen hat. Mit dieser Option wird nur das Protokoll SSL 3.0 unterstützt.

Zusätzlich zu den Konstanten SckTlsMax, SckTlsHigh, SckTlsMed und SckTlsLow kann mit SckTlsSNI beim Verbindungsauftbau die Server Name Indication (SNI) eingesetzt werden. Als Name wird der Hostname (alpha2) verwendet.

Zusätzlich kann in den Optionen MailAuthNTLM angegeben werden um eine Authentifizierung mittels NT LAN Manager (kurz NTLM) durchzuführen. Unterstützt der Mailserver dies, werden Benutzername (alpha5) und Passwort (alpha6) ignoriert und der aktuell am Computer angemeldete Domänenbenutzer verwendet.

Der Name oder die IP-Adresse des Mailservers in (alpha2) ist zwingend notwendig. (int3) muss nur dann angegeben werden, wenn der Mailserver auf einem anderen Port als auf Port 25 Mails entgegennimmt. Mit (int4) kann bestimmt werden, wie lange bei der Kommunikation mit dem Mailserver auf eine Antwort gewartet werden soll. Standardmäßig wird 60 Sekunden gewartet - ein höherer Wert sollte bei stark belasteten Mailservern benutzt werden, wenn beim Versenden ein Übertragungsfehler

Kontakt

auftritt. Der Timeout kann zwischen 10 und 300 Sekunden (5 Minuten) liegen.

Wird als Mailtyp (int1) MailCreateEML angegeben, wird die E-Mail nicht versendet. Statt dessen wird eine EML-Datei nach RFC 5322 erzeugt. Der Pfad und der Name der EML-Datei werden in (alpha2) angegeben. Alle weiteren Optionen werden ignoriert. Gespeicherte EML-Dateien können mit MailData() mit der Option MailFileEML wieder als Inhalt verwendet werden.

Sofern der verwendete Mailserver eine Authentifizierung erfordert, kann in (alpha5) der Benutzername (meist in der Form name@domain.tld) und in (alpha6) das Kennwort übergeben werden. Der Mailserver muss das SMTP-Kommando "AUTH" in Verbindung mit einer der Methoden "PLAIN", "LOGIN" oder "CRAM-MD5" unterstützen. Sofern der Server mehrere Methoden unterstützt, wird bevorzugt CRAM-MD5 verwendet, gefolgt von LOGIN.



MailOpen() generiert lediglich den Mailpuffer, es wird noch keine Verbindung zum Mailserver aufgebaut.

Das Resultat ist der neue Deskriptor, der anschließend bei MailData() und MailClose() verwendet wird.

Ist das Resultat ErrOutOfMemory, konnte kein Mailpuffer angelegt werden.

Kontakt

obj ->

MailData(int1,
alpha2[, alpha3[,
alpha4]]) : int



E-Mail definieren

obj Mail-Deskriptor

int1 Datentyp (siehe Text)

alpha2 Dateninhalt

alpha3 erweiterter Dateninhalt
(optional)

Objektreferenz /

"Delivery Status

alpha4 "Notification"-Kommandos
(optional)

Resultat int Ergebnis
Verwandte Befehle,

Siehe

MailOpen(), MailClose()

Mit dem Befehl MailData() werden der Nachrichtenkopf, der Nachrichteninhalt und die Anhänge gesetzt. In (obj) wird der durch MailOpen() erzeugte Deskriptor übergeben. (int1) legt fest, welche Daten gesetzt werden. In (alpha2), (alpha3) und (alpha4) stehen die zu setzenden Werte.

Nähere Informationen finden Sie in den einzelnen Abschnitten:

- Nachrichtenkopf definieren
- Nachrichteninhalt definieren
- Anhänge definieren

Der Befehl gibt folgende Fehlerwerte zurück:

Fehlerwert Bedeutung

0	ok, kein Fehler
13	Datentyp unzulässig
14	Datei nicht vorhanden
16	Format unzulässig
19	interner Fehler

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

obj ->



MailClose(int1[,

var alpha2]) : int

E-Mail versenden

obj Mail-Deskriptor

Verarbeitungsart

SmtpSendNow Nachricht

int1 versenden

SmtpDiscard

Nachricht

verwerfen

Letzte Antwort des

var

Mailservers bei

alpha2

Fehler (optional)

Resultat int Ergebnis



Verwandte Befehle,

Siehe MailOpen(),

MailData(), Blog

Mit diesem Befehl wird die aufbereitete Mail (obj) entweder abgesendet (int1 =

SmtpSendNow) oder verworfen (int1 = SmtpDiscard).

Wird beim Aufruf im optionalen Argument (alpha2) eine Variable zusammen mit der Option SmtpSendNow verwendet, wird bei Auftreten eines Fehlers die zuletzt erhaltene Antwort des Mail-Servers zurückgegeben. Der zurückgegebene Text ist leer, wenn entweder kein Fehler auftrat oder der Fehler nicht auf eine Antwort des Mail-Servers zurückgeht (z. B. Socket-Fehler).

Das Resultat informiert über den Erfolg des Mailversands:

Fehlerwert Bedeutung

0 (ErrOk) ok, kein Fehler - Mail wurde versendet

8 kein Empfänger bekannt

9 Absender fehlt

11 Keine Verbindung zum Mailserver

12 Verbindungsabbruch beim Übertragen

19 interner Fehler

ab 20 SMTP-Fehler:

20 - 29 Fehler beim Verbindungsauflauf

23 Service nicht verfügbar

27 Sonstiger Fehler

30 - 39 Fehler bei der Identifizierung

31 Syntaxfehler

32 Befehl oder Parameter unbekannt

33 Service nicht verfügbar

37 Sonstiger Fehler

38 Fehler bei der Authentifizierung

40 - 49 Fehler bei einer Empfangsadresse

Kontakt

41	Syntaxfehler
42	Befehl oder Parameter unbekannt
43	Service nicht verfügbar
44	Zuviele Daten
45	Interner Verarbeitungsfehler
46	Empfänger unzulässig
47	Sonstiger Fehler
50 - 59	<i>Fehler bei einer Absenderadresse</i>
51	Syntaxfehler
53	Service nicht verfügbar
54	Zuviele Daten
55	Interner Verarbeitungsfehler
57	Sonstiger Fehler
58	Fehler bei der Authentifizierung - Mailserver erzwingt Verschlüsselung

60 - 69 Fehler bei der Datenübergabe

61	Syntaxfehler
63	Service nicht verfügbar
64	Zuviele Daten
65	Interner Verarbeitungsfehler
66	Anhänge zu groß oder Absenderadresse aus fremder Domain ohne Relay
67	Sonstiger Fehler
70 - 79	<i>Fehler beim Verbindungsabbau</i>
71	Syntaxfehler
77	Sonstiger Fehler

Wenn eine EML-Datei geschrieben werden soll (siehe [MailCreateEML](#)), können zusätzlich Fehler für externe Dateioperationen auftreten.

Sollte die Verbindung zum Mailserver fehlschlagen, kann mit einem Telnet-Client überprüft werden, ob ein generelles Problem vorliegt:

```
telnet <Mailserveradresse> <Port>EHLOMAIL FROM:<Absenderadresse>RCPT TO:<Empfängeradresse>DATASub
```

Unterstützt der Mailserver kein ESMTMP, muss statt EHLO HELO angegeben werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Kontakt

Konstanten für E-Mail-Kopf

Konstanten für den E-Mail-Kopf

Siehe [E-Mail-Befehle](#)

- [SmtpBCC](#)
- [SmtpCC](#)
- [SmtpDiscard](#)
- [SmtpFrom](#)
- [SmtpPriority](#)
- [SmtpReplyTo](#)
- [SmtpSendNow](#)
- [SmtpSubject](#)
- [SmtpTo](#)

Kontakt

_SmtpBCC

Weitere Empfänger

Wert 61.444 /

0x0000F004

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - versteckten Empfänger der Kopie definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend die E-Mail-Adresse und der Realname des Empfängers einer Kopie der Mail angegeben, wobei diese Empfänger nicht im Mailheader vorkommen und daher für die anderen Empfänger nicht sichtbar sind.

Kontakt

_SsmtpCC

Weitere Empfänger

Wert 61.443 /

0x0000F003

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Empfänger der Kopie definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend die E-Mail-Adresse und der Realname des Empfängers einer Kopie der Mail angegeben.

Kontakt

_SmtpDiscard

Nachricht nicht versenden

Wert 1

Verwandte

Siehe [Befehle](#),

[MailClose\(\)](#)

Option bei [MailClose\(\)](#) - Nachricht nicht versenden.

Bei der Verwendung dieser Option wird bei [MailClose\(\)](#) die zuvor definierte Mail verworfen.

Kontakt

_SmtpFrom

Absender

Wert 61.441 /

0x0000F001

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Absender definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend die E-Mail-Adresse und der Realname des Absenders angegeben.

Kontakt

_SmtpNotifyTo

Empfangsbestätigungs-Adresse

Wert 61.448 /

0x0000F008

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Empfangs- bzw. Lesebestätigung an die angegebene Adresse anfordern.

Bei der Verwendung dieser Option wird bei MailData() eine Empfangsbestätigung angefordert. Die Bestätigungs-Mail wird an die angegebene Adresse versendet. Im E-Mail-Header wird der Eintrag Disposition-Notification-To gesetzt.



Die Anforderung einer Empfangsbestätigung bedeutet nicht, dass diese automatisch versendet wird. Einige E-Mail-Programme können so eingestellt werden, dass niemals eine Empfangsbestätigung versendet wird. Andere fragen den Empfänger, ob er eine Bestätigung versenden möchte.

Kontakt

_SmtpPriority

Priorität

Wert 61.697 /
0x0000F101

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Priorität definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend die Priorität der E-Mail angegeben.

Es sind fünf Prioritätsstufen möglich:

Priorität Bedeutung

- 1 Höchste Priorität
- 2 Hohe Priorität
- 3 Normale Priorität
- 4 Niedrige Priorität
- 5 Niedrigste Priorität

Nicht alle Mailserver werten die Priorität einer Mail aus, somit kann eine bevorzugte Beförderung der Mail nicht garantiert werden.

Standardmäßig wird die Prioritätsstufe 3 verwendet.

Kontakt

_SmtpReplyTo

Antwortadresse

Wert 61.446 /
0x0000F006

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Antwortadresse definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend die E-Mail-Adresse und der Realname des Empfängers einer Antwort angegeben. Die Adresse wird bei der Verwendung von "Verfasser antworten" als Zieladresse verwendet.

Kontakt

_SmtpSendNow

Nachricht versenden

Wert 0

Verwandte

Siehe [Befehle](#),

[MailClose\(\)](#)

Option bei [MailClose\(\)](#) - Nachricht versenden.

Bei der Verwendung dieser Option wird bei [MailClose\(\)](#) die zuvor definierte Mail abgeschickt.

Kontakt

_SmtpSubject

Betreff

Wert 61.445 /
0x0000F005

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Betreff definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend der Betreff der Nachricht angegeben. Die Länge des Betreffs ist auf 80 Zeichen beschränkt.

Kontakt

_SmtpTo

Empfänger

Wert 61.442 /

0x0000F002

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Empfänger definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend die E-Mail-Adresse und der Realname des Empfängers angegeben.

Kontakt

_SmtpXtension

Nicht standadisierter Headereintrag

Wert 61.953 /

0x0000F201

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Nicht standardisierten Headereintrag definieren.

Bei der Verwendung dieser Option wird bei MailData() anschließend der Name des Headereintrags und der Wert des Eintrags angegeben.



Der Name des nicht standardisierten Headereintrags muss mit "X-" beginnen, sonst enthält die E-Mail den Headereintrag nicht.

Oftmals werden diese Headereinträge von Spamfiltern und Virensuchern oder dem E-Mail-Client des Absenders hinterlegt. Sie werden in der Regel nicht vom E-Mail-Client ausgewertet.

Beispiele:

```
// Normale PrioritätX-Priority: 3// E-Mail-Client des AbsendersX-Mailer: Microsoft Outlook Expr
```

Kontakt

Konstanten für E-Mail-Körper

Konstanten für den E-Mail-Körper

Siehe [E-Mail-Befehle](#)

- [MailBuffer](#)
- [MailFile](#)
- [MailFileEML](#)
- [MailLine](#)
- [MimeTextHtml](#)

Kontakt

_MailBuffer

Textpuffer einfügen

Wert 393.220 /

0x00060004

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Nachrichtenkörper um Textpuffer erweitern.

Bei der Verwendung dieser Option wird bei MailData() anschließend der nach alpha konvertierte Texthandle (CnvAI()) angegeben.

Kontakt

_MailFile

Externe Datei einfügen

Wert
0 /

0x00000000

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - externe Datei als Nachrichtenkörper verwenden.

Bei der Verwendung dieser Option wird bei MailData() anschließend der Name der externen Datei und der MIME-Typ angegeben.

_MailFileEML

Externe EML-Datei einfügen

Wert 53.248 /
0x0000D000

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - externe EML-Datei als Nachrichtenkörper verwenden.

Bei der Verwendung dieser Option wird bei MailData() anschließend der Name der externen EML-Datei und der MIME-Typ angegeben.

-  Wird eine EML-Datei als Nachrichteninhalt definiert, kann kein weiterer Inhalt oder Anhang angefügt werden.

Kontakt

_MailLine

Textzeile einfügen

Wert 131.076 /

0x00020004

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Nachrichtenkörper um Textzeile erweitern.

Bei der Verwendung dieser Option wird bei MailData() anschließend die Textzeile angegeben, um die der Nachrichtenkörper erweitert werden soll.

Kontakt

_MimeTextHtml

HTML-Textzeile

Wert 12.288 /

0x00003000

Verwandte

Siehe [Befehle](#),

[MailData\(\)](#)

Option bei [MailData\(\)](#) - Hinzufügen einer Textzeile im HTML-Format.

Diese Option kann zusammen mit [MailLine](#) angegeben werden. Damit wird eine HTML-Zeile an dem Körper der E-Mail angehängt.

Konstanten für E-Mail-Anhänge

Konstanten für die E-Mail-Anhänge

Siehe [E-Mail-Befehle](#)

- [MimeApp](#)
- [MimeAppMSWORD](#)
- [MimeAppPDF](#)
- [MimeAppPS](#)
- [MimeAppRTF](#)
- [MimeAppZIP](#)
- [MimeContentID](#)
- [MimeCS IBM437](#)
- [MimeCS IBM850](#)
- [MimeCS ISO 8859 1](#)
- [MimeCS UTF8](#)
- [MimeImageGIF](#)
- [MimeImageJPEG](#)
- [MimeImagePNG](#)
- [MimeImageTIFF](#)
- [MimeOther8B](#)
- [MimeOtherB64](#)
- [MimeOtherQP](#)
- [MimeRelated](#)
- [MimeTE 8B](#)
- [MimeTE B64](#)
- [MimeTE QP](#)

Kontakt

_MimeApp

Allgemeinen Datentyp anhängen

Wert 8.450 /

0x00002102

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang von allgemeinem Datentyp.

Mit dieser Option wird beim Befehl MailData() der Anhang als Datei von einem allgemeinen Datentyp deklariert.

Kontakt

_MimeAppMSWORD

Microsoft Word-Dokument anhängen

Wert 9.730 /

0x00002602

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang ist vom Typ DOC.

Mit dieser Option wird beim Befehl MailData() der Anhang als Microsoft Word-Dokument deklariert.

Kontakt

_MimeAppPDF

PDF-Dokument anhängen

Wert 9.218 /

0x00002402

Verwandte

Siehe [Befehle](#),

[MailData\(\)](#)

Option bei [MailData\(\)](#) - Anhang ist vom Typ PDF.

Mit dieser Option wird beim Befehl [MailData\(\)](#) der Anhang als PDF-Datei deklariert.

Kontakt

_MimeAppPS

Postscript-Datei anhängen

Wert 8.706 /

0x00002202

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang ist vom Typ PS.

Mit dieser Option wird beim Befehl MailData() der Anhang als Postscript-Datei deklariert.

Kontakt

_MimeAppRTF

RTF-Dokument anhängen

Wert 8.962 /

0x00002302

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang ist vom Typ RTF.

Mit dieser Option wird beim Befehl MailData() der Anhang als RTF-Datei deklariert.

Kontakt

_MimeAppZIP

ZIP-Archiv anhängen

Wert 9.474 /

0x00002502

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang ist vom Typ ZIP.

Mit dieser Option wird beim Befehl MailData() der Anhang als ZIP-Datei deklariert.

Kontakt

_MimeContentID

Eingebetteter Anhang über Content-ID

Wert 2.097.152 /

0x00200000

Verwandte

Siehe Befehle,

MailData(),

MimeRelated

Option bei MailData() - Anhang wird über eine Content-ID in den E-Mail-Body eingebettet.

Diese Option muss zusammen mit MimeRelated angegeben werden.

Kontakt

_MimeCS_IBM437

IBM-Zeichensatz

Wert 16 /

0x00000010

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Verwendung des IBM Zeichensatzes 437.

Zur Darstellung des Mailtextes wird der IBM Zeichensatz 437 verwendet.

Kontakt

_MimeCS_IBM850

IBM-Zeichensatz

Wert 32 /

0x00000020

Verwandte

Siehe **Befehle**,

MailData()

Option bei **MailData()** - Verwendung des IBM Zeichensatzes 850.

Zur Darstellung des Mailtextes wird der IBM Zeichensatz 850 verwendet.

Kontakt

_MimeCS_ISO_8859_1

ISO-Zeichensatz

Wert

0 / 0x00000000

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Verwendung des ISO-8859-1 Zeichensatzes.

Zur Darstellung des Mailtextes wird der Zeichensatz ISO-8859-1 verwendet.

Kontakt

_MimeCS_UTF8

Unicode-Zeichensatz
240 /
Wert

0x000000F0

Verwandte

Siehe [Befehle](#),

[MailData\(\)](#)

Option bei [MailData\(\)](#) - Verwendung des UTF-8 Zeichensatzes.

Zur Darstellung des Mailtextes wird der Unicode-Zeichensatz UTF-8 verwendet.

Beispiele:

```
// Einzelne Zeile umwandelnt Mail->MailData(_MailLine | _MimeTE_QP | _MimeCS_UTF8, StrCnv(StrCnv('
```

Kontakt

_MimeImageGIF

GIF-Bild anhängen

Wert 4.354 /

0x00001102

Verwandte

Siehe [Befehle](#),

[MailData\(\)](#)

Option bei [MailData\(\)](#) - Anhang ist vom Typ GIF.

Mit dieser Option wird beim Befehl [MailData\(\)](#) der Anhang als GIF-Datei deklariert.

Kontakt

_MimeImageJPEG

JPG-Bild anhängen

Wert 4.610 /
0x00001202

Verwandte

Siehe **Befehle**,

MailData()

Option bei **MailData()** - Anhang ist vom Typ JPG.

Mit dieser Option wird beim Befehl **MailData()** der Anhang als JPEG-Datei deklariert.

Kontakt

_MimeImagePNG

PNG-Bild anhängen

Wert 5.122 /

0x00001402

Verwandte

Siehe **Befehle**,

MailData()

Option bei **MailData()** - Anhang ist vom Typ PNG.

Mit dieser Option wird beim Befehl **MailData()** der Anhang als PNG-Datei deklariert.

Kontakt

_MimeImageTIFF

TIF-Bild anhängen

Wert 4.866 /
0x00001302

Verwandte

Siehe **Befehle**,

MailData()

Option bei **MailData()** - Anhang ist vom Typ TIF.

Mit dieser Option wird beim Befehl **MailData()** der Anhang als TIFF-Datei deklariert.

_MimeOther8B

8-Bit kodierter Anhang

Wert 57.350 /

0x0000E006

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang wird nach 8-Bit kodiert.

Mit dieser Option wird beim Befehl MailData() der Anhang als beliebiger Typ deklariert.

Die Kodierung findet nach dem 8-Bit-Verfahren statt.

Bei der Verwendung dieser Option, muss als Parameter (alpha4) von MailData() ein gültiger Typ angegeben werden. Jeder Typ besteht aus einem Haupttyp und einem Untertyp. Derzeit sind folgende Haupttypen definiert:

- text
- multipart
- message
- application
- image
- audio
- video
- model

Zu jedem Haupttyp gibt es eine Anzahl verschiedener Untertypen. Registrierte Untertypen werden bei der IANA eingetragen (<http://www.iana.org>), unregistrierte Untertypen beginnen mit 'x-' (zum Beispiel audio/x-wav).

_MimeOtherB64

Base64 kodierter Anhang

Wert 57.346 /
0x0000E002

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang wird nach Base64 kodiert.

Mit dieser Option wird beim Befehl MailData() der Anhang als beliebiger Typ deklariert.

Die Kodierung findet nach dem Base64-Verfahren statt.

Bei der Verwendung dieser Option, muss als Parameter (alpha4) von MailData() ein gültiger Typ angegeben werden. Jeder Typ besteht aus einem Haupttyp und einem Untertyp. Derzeit sind folgende Haupttypen definiert:

- text
- multipart
- message
- application
- image
- audio
- video
- model

Zu jedem Haupttyp gibt es eine Anzahl verschiedener Untertypen. Registrierte Untertypen werden bei der IANA eingetragen (<http://www.iana.org>), unregistrierte Untertypen beginnen mit 'x-' (zum Beispiel audio/x-wav).

_MimeOtherQP

Quoted-printable kodierter Anhang

Wert 57.348 /
0x0000E004

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang wird nach quoted-printable kodiert.

Mit dieser Option wird beim Befehl MailData() der Anhang als beliebiger Typ deklariert.

Die Kodierung findet nach dem quoted-printable-Verfahren statt.

Bei der Verwendung dieser Option, muss als Parameter (alpha4) von MailData() ein gültiger Typ angegeben werden. Jeder Typ besteht aus einem Haupttyp und einem Untertyp. Derzeit sind folgende Haupttypen definiert:

- text
- multipart
- message
- application
- image
- audio
- video
- model

Zu jedem Haupttyp gibt es eine Anzahl verschiedener Untertypen. Registrierte Untertypen werden bei der IANA eingetragen (<http://www.iana.org>), unregistrierte Untertypen beginnen mit 'x-' (zum Beispiel audio/x-wav).

Kontakt

_MimeRelated

Eingebetteter Anhang

Wert 1.048.576 /
0x00100000

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Anhang wird in den E-Mail-Body eingebettet.

Kontakt

_MimeTE_8B

Unkodierter Nachrichteninhalt
Wert 6 /

0x00000006

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - keine Kodierung.

Mit dieser Option wird beim Befehl MailData() der Nachrichteninhalt nicht kodiert.

Kontakt

_MimeTE_B64

Base64 Kodierung

Wert

0x00000002

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Kodierung nach Base64.

Mit dieser Option wird beim Befehl MailData() der Nachrichteninhalt nach Base64 kodiert.

Kontakt

_MimeTE_QP

Quoted-printable Kodierung

Wert
4 /

0x00000004

Verwandte

Siehe Befehle,

MailData()

Option bei MailData() - Kodierung nach quoted-printable.

Mit dieser Option wird beim Befehl MailData() der Nachrichteninhalt nach quoted-printable kodiert.

Kontakt

Befehle für ODBC-Verbindungen

Liste der Befehle zur Verbindung mit anderen Datenbanken über ODBC

Schnittstellen
und
Kommunikation,
Siehe Befehlsgruppen,
Alphabetische
Liste aller
Befehle

Mit den hier beschriebenen Anweisungen und Objekten kann aus CONZEPT 16 heraus auf eine beliebige ODBC-Datenquelle zugegriffen werden. Die Beschreibung des Zugriffs von beliebigen Applikationen auf eine CONZEPT 16-Datenbank als Datenquelle befindet sich im Abschnitt Die ODBC-Schnittstelle - Der ODBC-Treiber. In der Beispieldatenbank "Codelibrary" befindet sich ein Beispiel "ODBC" mit dem der Zugriff auf eine beliebige Datenquelle ermöglicht wird.

Objekte der ODBC-Schnittstelle

Die ODBC-Schnittstelle wird mit der Anweisung OdbcOpen() initialisiert. Die Anweisung gibt einen Deskriptor auf das OdbcApi-Objekt zurück. Das Objekt wird beim Verbinden mit einer Datenquelle benötigt. Ausgehend von dem Objekt kann eine Verbindung zu einer Datenquelle mit der Anweisung OdbcConnect() oder OdbcConnectDriver() hergestellt werden. Es ist möglich mehrere Verbindungen zu unterschiedlichen Datenquellen herzustellen.

```
@A+@C+mainlocal{ tOdbcApi      : handle;  tOdbcCon      : handle;  tOdbcStmt      : handle;  tcus
```

Zugriff auf Daten in einer Tabelle

Nachdem eine Verbindung zu einer Datenquelle hergestellt ist, können Daten aus der Quelle gelesen werden. Im einfachsten Fall wird ein ODBC-Statement zur Datenquelle geschickt (OdbcExecuteDirect()) und das Resultat ausgewertet:

```
tOdbcStmt # tOdbcCon->OdbcExecuteDirect('SELECT * FROM KND_D_Kunden'); while (tOdbcStmt->OdbcF
```

In diesem Beispiel werden alle Spalten aller Datensätze der Tabelle "KND_D_Kunden" gelesen. Mit der Anweisung OdbcFetch() wird auf eine Zeile des Ergebnisses positioniert. In der Schleife wird mit OdbcClmData() der Inhalt der Spalten einzelnen Variablen zugeordnet. Wird das OdbcStm-Objekt nicht mehr benötigt, wird es mit OdbcClose() entfernt.

Bei der Abfrage der Daten werden die Werte in den Spalten automatisch in den Datentyp der übergebenen Variablen gewandelt. Ist eine Wandlung nicht möglich, wird die Variable auf 0 gesetzt. Der Typ der Spalte kann mit der Anweisung OdbcClmInfo() ermittelt werden. Die Anzahl der vom ODBC-Statement zurückgegebenen Spalten befindet sich in der Eigenschaft OdbcResCountClm.

Schließen der ODBC-Verbindung und Entfernen der Objekte

Das Schließen von Objekten der ODBC-Schnittstelle erfolgt mit der Anweisung OdbcClose().

Kontakt

```
... tOdbcCon->OdbcClose(); tOdbcApi->OdbcClose();}
```

Ermitteln von Fehlerwerten und -texten

In dem bisher vorgestellten Beispiel wurde aus Gründen der Übersichtlichkeit auf eine Fehlerbehandlung verzichtet. Fehler werden entweder von den CONZEPT 16-Anweisungen zurückgegeben oder von dem verbundenen ODBC-Treiber erzeugt.

Tritt ein Fehler beim ODBC-Treiber auf, liefert die CONZEPT 16-Anweisung ErrOdbcError zurück. Der Fehler des ODBC-Treibers kann über die Eigenschaften OdbcErrMsg und OdbcErrSqlNativeCode des übergeordneten Objekts ermittelt werden.

```
... tOdbcStmt # tOdbcCon->OdbcExecuteDirect('SELECT * FROM KND_D_Kunden'); if (tOdbcStmt <= 0
```

Ermitteln von Informationen über die Datenquellen und aus den Datenquellen

Nach der Initialisierung der ODBC-Schnittstelle, können die definierten ODBC-Datenquellen mit der Anweisung OdbcDataSources() ermittelt werden. Die Anweisung gibt eine Cte-Liste mit den Datenquellen zurück. In der Eigenschaft Name der CteItem-Objekte befinden sich die Namen der Datenquellen.

Ist eine Verbindung zu einer Datenquelle hergestellt, stehen über das OdbcCon-Objekt bereits einige Informationen zur Verfügung (siehe Eigenschaften eines OdbcCon-Objekts). Sollen die Tabellen und deren Spalten ermittelt werden, müssen die Anweisungen OdbcCatalogTbl() und OdbcCatalogClm() verwendet werden.

```
@A+@C+mainlocal{ tOdbcApi : handle; tOdbcCon : handle; tOdbcTbl : handle; tOdbcClm : handle;
```

Diese Funktion ermittelt für alle Tabellen die zugehörigen Spaltennamen. Über das entsprechende Spalten-Objekt kann ebenfalls der Datentyp und weitere Informationen der Spalte ermittelt werden (siehe Eigenschaften eines OdbcClm-Objekts).

Befehle für ODBC-Verbindungen

- OdbcCatalogClm
- OdbcCatalogTbl
- OdbcClmData
- OdbcClmInfo
- OdbcClose
- OdbcConnect
- OdbcConnectDriver
- OdbcDataSources
- OdbcExecute
- OdbcExecuteDirect
- OdbcFetch
- OdbcOpen
- OdbcParamAdd
- OdbcParamSet
- OdbcPrepare

Befehle für die Startprozedur

Kontakt

- FldAttributes
- ProcAdvertise

Kontakt

OdbcOpen() : handle  Initialisieren der
ODBC-Schnittstelle

Resultat handle Deskriptor auf ein ODBC-Schnittstellen-Objekt

Siehe Verwandte Befehle, OdbcApi, OdbcClose()

Mit dieser Anweisung wird die ODBC-Schnittstelle initialisiert. Der Aufruf muss vor der Verwendung aller anderen ODBC-Befehle oder -Eigenschaften erfolgen. Wurde die API erfolgreich initialisiert, wird ein Deskriptor auf ein OdbcApi-Objekt zurückgegeben. Über diesen Deskriptor können folgende Anweisungen aufgerufen werden:

- OdbcClose()

Die Schnittstelle wird wieder beendet. Die angeforderten Ressourcen werden freigegeben.

- OdbcDataSources()

Die vorhandenen Datenquellen werden ermittelt und stehen anschließend in einem CteList-Objekt zur Verfügung.

- OdbcConnect()

Es wird eine Verbindung zu einer Datenquelle aufgebaut.

- OdbcClose()

Die Verbindung zu einer Datenquelle wird getrennt.

Konnte die Schnittstelle nicht initialisiert werden, wird einer der folgenden Fehlerwerte zurückgegeben:

ErrOutOfMemory

Der Hauptspeicher reicht nicht aus.

ErrOdbcIncomplete

Es konnten nicht alle notwendigen ODBC-Funktionen initialisiert werden.

ErrOdbcEnvironment

Die ODBC-Umgebung konnte nicht initialisiert werden.

ErrOdbcNotFound

Die dynamische Linkbibliothek odbc32.dll konnte nicht geladen werden.

Kontakt

obj -> OdbcClose()  Schließen von 

ODBC-Objekten

Deskriptor eines
obj

ODBC-Objekts

Verwandte Befehle,

OdbcOpen(),

Siehe **OdbcConnect(),**

OdbcPrepare(),

OdbcExecuteDirect()

Mit dieser Anweisung werden Objekte, die mit den folgenden Anweisungen erstellt wurden, wieder entfernt:

- **OdbcOpen()**

Das **OdbcApi**-Objekt wird geschlossen. Sollten zu diesem Zeitpunkt noch Verbindungen zu Datenquellen bestehen, werden diese ebenfalls geschlossen.

- **OdbcConnect() oder OdbcConnectDriver()**

Das **OdbcCon**-Objekt wird geschlossen. Sollten zu diesem Zeitpunkt noch ODBC-Tabellen-, ODBC-Spalten- und ODBC-Statement-Objekte offen sein, werden diese ebenfalls geschlossen.

- **OdbcPrepare()**

Das **OdbcStm**-Objekt wird geschlossen. Eine erneute Durchführung des vorbereiteten ODBC-Statements ist anschließend nicht mehr möglich. Zudem kann nicht mehr auf die Ergebnismenge oder den Fehlerwert der vorangegangenen Ausführung zugegriffen werden.

- **OdbcExecuteDirect()**

Das **OdbcStm**-Objekt wird geschlossen. Ein Zugriff auf die Ergebnismenge oder den Fehlerwert der vorangegangenen Ausführung ist nicht mehr möglich.

Mögliche Laufzeitfehler:

ErrHdIInvalid Bei dem übergebenen Deskriptor handelt es sich nicht um ein gültiges ODBC-Objekt.

Kontakt

obj -> OdbcDataSources([int1])
: handle Datenquellen



lesen
obj Deskriptor des OdbcApi-Objekts
Optionen (optional)

_OdbcDataSourcesSystem System-Datenquellen

ermitteln (System)

_OdbcDataSourcesUser Benutzer-Datenquellen

ermitteln (User)

int1 _OdbcDataSourcesTree sortierte Liste

zurückgeben

_OdbcDataSourcesTreeCI sortierte Liste ohne Berücksichtigung der Groß-/Kleinschreibung

zurückgeben

Resultat handle Deskriptor auf ein Cte-Objekt

Siehe Verwandte Befehle, Dynamische Strukturen

Die Anweisung ermittelt die im ODBC-Datenquellen-Administrator eingetragenen Datenquellen. Dazu muss in (obj) der von OdbcOpen() zurückgegebene Deskriptor angegeben werden. Werden in (int1) keine weiteren Optionen angegeben, wird ein Deskriptor auf ein CteList-Objekt mit allen Datenquellen zurückgegeben.

- _OdbcDataSourcesSystem

Es werden nur die System-Datenquellen zurückgegeben. Die Option kann mit _OdbcDataSourcesUser kombiniert werden, um alle Datenquellen zu ermitteln.

- _OdbcDataSourcesUser

Es werden nur die Benutzer-Datenquellen zurückgegeben. Die Option kann mit _OdbcDataSourcesSystem kombiniert werden, um alle Datenquellen zu ermitteln.

- _OdbcDataSourcesTree

Anstelle einer Liste wird eine sortierte Liste zurückgegeben. Die Option ist nicht mit _OdbcDataSourcesTreeCI kombinierbar.

- _OdbcDataSourcesTreeCI

Anstelle einer Liste wird eine sortierte Liste ohne Berücksichtigung der Groß-/Kleinschreibung zurückgegeben. Die Option ist nicht mit _OdbcDataSourcesTree kombinierbar.

In der zurückgegebenen Liste ist für jede Datenquelle ein CteItem-Objekt vorhanden.

In der Eigenschaft Name ist der Name der Datenquelle, in der Eigenschaft Custom die Beschreibung enthalten.



Unter 64-Bit-Systemen befindet sich die relevante "Microsoft ODBC Administrator" Anwendung im Verzeichnis Windows\SysWOW64\ unter dem Namen odbcad32.exe.

Kontakt

Steht zur Ausführung der Anweisung nicht genügend Hauptspeicher zur Verfügung, wird der Fehler ErrOutOfMemory zurückgegeben.

Beispiele:

Ermitteln einer unsortierten Liste aller Datenquellen:

```
tOdbcApi # OdbcOpen() ; tOdbcSourceList # tOdbcApi->OdbcDataSources() ; for      tOdbcSource # tOdbcSour
```

Ermitteln aller System-Datenquellen nach Namen sortiert (ohne Berücksichtigung der Groß-/Kleinschreibung):

```
tOdbcApi # OdbcOpen() ; tOdbcSourceList # tOdbcApi->OdbcDataSources(_OdbcDataSourcesSystem | _OdbcD
```

Mögliche Laufzeitfehler:

Im (obj) wurde kein gültiger Deskriptor auf ein OdbcApi-Objekt ErrHdlInvalid übergeben.

Kontakt

obj -> OdbcConnect(alpha1[, alpha2[, alpha3]]) : handle



Verbindung zu einer Datenquelle herstellen

obj Deskriptor der ODBC-Schnittstelle

alpha1 Name der Datenquelle

alpha2 Benutzer (optional)

alpha3 Kennwort (optional)

Resultat handle Deskriptor des Verbindungsobjekts

Verwandte Befehle, OdbcOpen(),

Siehe

OdbcConnectDriver(), OdbcClose()

Mit dieser Anweisung wird eine Verbindung zu einer Datenquelle hergestellt. Als (obj) wird ein Deskriptor auf ein gültiges OdbcApi-Objekt übergeben (siehe OdbcOpen()). Der Name der Datenquelle wird in (alpha1) übergeben. Die vorhandenen Datenquellen können zuvor mit der Anweisung OdbcDataSources() ermittelt werden.

Ist eine Benutzeranmeldung notwendig, muss in (alpha2) und (alpha3) der Benutzer und das Kennwort angegeben werden.

Bei erfolgreicher Durchführung liefert der Befehl einen Deskriptor auf ein OdbcCon-Objekt zurück. Dieser Deskriptor kann zum Aufruf weiterer Befehle verwendet werden:

- OdbcCatalogTbl()
- OdbcCatalogClm()
- OdbcExecuteDirect()
- OdbcPrepare()
- OdbcClose()

Die Verbindung zur Datenquelle wird mit der Anweisung OdbcClose() wieder getrennt.

Es können Verbindungen zu mehreren Datenquellen gleichzeitig aufgebaut werden.

Konnte keine Verbindung zu der angegebenen Datenquelle aufgebaut werden, gibt die Anweisung den Fehlerwert ErrOdbcError zurück. Genauere Informationen über den Fehler können über die OdbcErr...-Eigenschaften des OdbcApi-Objekts abgefragt werden. Ist schon die Erstellung des ODBC-Verbindungs-Objekts fehlgeschlagen, gibt die Anweisung ErrOdbcFunctionFailed zurück.

Mögliche Laufzeitfehler:

ErrHdliInvalid Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges OdbcApi-Objekt.

Kontakt

obj -> OdbcConnectDriver(alpha1) : handle |

Verbindung zu einer Datenquelle herstellen

obj Deskriptor der ODBC-Schnittstelle

alpha1 Verbindungszeichenfolge

Resultat handle Deskriptor des Verbindungsobjekts

Siehe Verwandte Befehle, OdbcOpen(),

OdbcConnect(), OdbcClose()

Mit dieser Anweisung wird eine Verbindung zu einer Datenquelle hergestellt. Als (obj) wird ein Deskriptor auf ein gültiges OdbcApi-Objekt übergeben (siehe OdbcOpen()). Die Verbindungszeichenfolge wird in (alpha1) übergeben. Die Zeichenfolge ist abhängig vom angesprochenen ODBC-Treiber.

Die Verbindungszeichenfolge kann über den Dialog WinComOdbc erstellt werden.

Die Eigenschaft ConnectionString enthält die entsprechende Zeichenfolge.

Bei erfolgreicher Durchführung liefert der Befehl einen Deskriptor auf ein OdbcCon-Objekt zurück.

Die Verbindung zur Datenquelle wird mit der Anweisung OdbcClose() wieder getrennt.

Es können Verbindungen zu mehreren Datenquellen gleichzeitig aufgebaut werden.

Konnte keine Verbindung zu der angegebenen Datenquelle aufgebaut werden, gibt die Anweisung den Fehlerwert ErrOdbcError zurück. Genauere Informationen über den Fehler können über die OdbcErr...-Eigenschaften des Verbindungs-Objekts abgefragt werden. Ist schon die Erstellung des ODBC-Verbindungs-Objekts fehlgeschlagen, gibt die Anweisung ErrOdbcFunctionFailed zurück.

Mögliche Laufzeitfehler:

ErrHdliInvalid Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges OdbcApi-Objekt.

Kontakt

obj -> OdbcCatalogTbl([alpha1[,

alpha2[, alpha3[, alpha4]]]]) :

handle

Tabelleninformationen ermitteln

Deskriptor eines

obj

ODBC-Verbindungs-Objekt

alpha1 Name der Tabelle

alpha2 Tabellen-Typ

alpha3 Name des Katalogs

alpha4 Name des Schemas

Deskriptor auf ein

Resultat handle

ODBC-Tabellen-Objekt

Siehe

Verwandte Befehle, OdbcConnect(),

OdbcClose()

Mit der Anweisung werden die Informationen zu einer oder allen Tabellen in der Datenquelle ermittelt. Der Deskriptor des ODBC-Verbindungs-Objekts (OdbcCon) wird als (obj) übergeben (siehe OdbcConnect()). Bei erfolgreicher Durchführung wird ein Deskriptor auf ein OdbcTbl-Objekt zurückgegeben.

Im Fehlerfall gibt der Befehl den Wert ErrOdbcError. Genauere Informationen können über die Fehler-Eigenschaften (OdbcErr...) des ODBC-Verbindungs-Objekts abgefragt werden. Ist schon die Erstellung des ODBC-Tabellen-Objekts fehlgeschlagen, gibt die Anweisung ErrOdbcFunctionFailed zurück.

Ohne weitere Übergabeparameter werden alle Tabellen der Datenquelle ermittelt. Mit der Anweisung OdbcFetch() können die Tabellen nacheinander gelesen werden. Ist der Name der zu ermittelnde Tabelle bekannt, kann dieser in (alpha1) direkt angegeben werden. Sofern Tabellen-Typ, Katalogname oder der Schema-Name bekannt sind, können diese in den Argumenten (alpha2) bis (alpha4) angegeben werden.

Das Objekt bleibt erhalten, bis die Verbindung zur Datenquelle getrennt, oder das Objekt selbst mit OdbcClose() geschlossen wird.

Beispiel:

Ermitteln aller Tabellen in der Datenquelle:

```
tOdbcApi # OdbcOpen(); tOdbcCon # tOdbcApi->OdbcConnect('CRM', 'user', ''); tOdbcCatTbl # tOdbcCon-
```

Mögliche Laufzeitfehler:

ErrHdIInvalid Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges OdbcCon-Objekt.



Kontakt

obj ->

OdbcCatalogClm([alpha1[,
alpha2[, alpha3[, alpha4]]]]):
handle



Spalteninformationen ermitteln
Deskriptor auf ein
obj

ODBC-Verbindungs-Objekt

alpha1 Name der Tabelle

alpha2 Name der Spalte

alpha3 Name des Katalogs

alpha4 Name des Schemas

Deskriptor auf ein

Resultat handle

ODBC-Spalten-Objekt

Siehe [Verwandte Befehle, OdbcConnect\(\)](#),

[OdbcClose\(\)](#)

Mit der Anweisung werden die Informationen zu einer oder allen Spalten einer Tabelle oder der Datenquelle ermittelt. Der Deskriptor des ODBC-Verbindungs-Objekts ([OdbcCon](#)) wird als (obj) übergeben (siehe [OdbcConnect\(\)](#)). Bei erfolgreicher Durchführung wird ein Deskriptor auf ein [OdbcClm](#)-Objekt zurückgegeben.

Im Fehlerfall gibt der Befehl den Wert [ErrOdbcError](#). Genauere Informationen können über die Fehler-Eigenschaften ([OdbcErr...](#)) des ODBC-Verbindungs-Objekts abgefragt werden. Ist schon die Erstellung des ODBC-Spalten-Objekts fehlgeschlagen, gibt die Anweisung [ErrOdbcFunctionFailed](#) zurück.

Ohne weitere Übergabeparameter werden alle Spalten der Datenquelle ermittelt. Sollen alle Spalten einer Tabelle ermittelt werden, muss der Name der Tabelle in (alpha1) übergeben werden. Durch die Angabe des Spaltennamens in (alpha2) kann die Auswahl weiter eingeschränkt werden. In (alpha3) und (alpha4) kann ein abweichender Katalogname und ein Schemaname angegeben werden.

Mit der Anweisung [OdbcFetch\(\)](#) können die Spalten nacheinander gelesen werden.

Das Objekt bleibt erhalten, bis die Verbindung zur Datenquelle getrennt, oder das Objekt selbst mit [OdbcClose\(\)](#) geschlossen wird.

Beispiel:

Ermitteln aller Spalten einer Tabelle:

```
tOdbcApi # OdbcOpen(); tOdbcCon # tOdbcApi->OdbcConnect('CRM', 'user', ''); tOdbcCatClm # tOdbcCon-
```

Mögliche Laufzeitfehler:

Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges
[ErrHdlInvalid](#) [OdbcCon](#)-Objekt.

Kontakt

obj ->
OdbcPrepare(alpha1) : handle
SQL-Statement vorbereiten
Deskriptor eines
obj ODBC-Verbindungs-Objekts
alpha1 SQL-Statement
Deskriptor auf ein
Resultat handle ODBC-Statement-Objekt



Siehe [OdbcConnect\(\)](#), [OdbcExecute\(\)](#),
[OdbcParamAdd\(\)](#)

Mit dieser Anweisung wird ein SQL-Statement zur Durchführung mit [OdbcExecute\(\)](#) vorbereitet. Das vorbereitete Statement kann mehrfach ausgeführt werden, ohne es erneut angeben zu müssen. Das wird besonders im Zusammenhang mit dem parametrisierten Aufruf (siehe [OdbcParamAdd\(\)](#)) nützlich.

Im Parameter (obj) wird ein Deskriptor auf ein [OdbcCon](#)-Objekt übergeben. In (alpha1) ist das SQL-Statement angegeben, welches für die Durchführung vorbereitet werden soll. Bei erfolgreicher Durchführung der Anweisung wird ein Deskriptor auf ein [OdbcStm](#)-Objekt zurückgegeben. Im Fehlerfall wird [ErrOdbcError](#) zurückgegeben. Weitere Informationen zu dem Fehler können dann über die [OdbcErr...-Eigenschaften](#) des ODBC-Verbindungs-Objekt ermittelt werden. Ist schon die Erstellung des ODBC-Statement-Objekts fehlgeschlagen, gibt die Anweisung [ErrOdbcFunctionFailed](#) zurück.

Je nach SQL-Statement sind zwei Fälle zu unterscheiden:

1. SQL-Abfrage (SELECT)

Nach der Durchführung des Befehls muss zunächst [OdbcExecute\(\)](#) ausgeführt werden. Anschliessend kann die Ergebnismenge durch mehrfache Aufrufe von [OdbcFetch\(\)](#) ermittelt werden.

2. Schreibendes SQL-Statement (z. B. INSERT)

Da es keine Ergebnismenge gibt, ist die Durchführung mit [OdbcExecute\(\)](#) abgeschlossen.

Das Objekt bleibt erhalten, bis die Verbindung zur Datenquelle getrennt, oder das Objekt selbst mit [OdbcClose\(\)](#) geschlossen wird.

Beispiel:

```
tOdbcStm # tOdbcCon->OdbcPrepare('SELECT Name FROM Customer');if (tOdbcStm > 0){ tOdbcStm->OdbcE
```

Kontakt

obj -> OdbcParamSet(int1, var2) : int           **Parameter**

für ODBC-Statement übergeben

Deskriptor des
obj ODBC-Statement-Objekts

int1 Parameterposition

var2 Parameterwert

Resultat int Fehlerwert

Siehe Verwandte Befehle,
OdbcPrepare(),
OdbcParamAdd()

Die Anweisung wird in Zusammenhang mit OdbcParamAdd() verwendet, um den Wert eines externen Statement-Parameters zuzuweisen.

In (obj) muss ein Deskriptor auf ein OdbcStm-Objekt übergeben werden. (int1) definiert den Index des Parameters, deren Wert gesetzt werden soll. Das Argument (var2) definiert den Wert des Parameters.

Konnte die Anweisung ausgeführt werden, wird [ErrOk](#) zurückgegeben. Stimmen der übergebene Datentyp nicht mit dem in [OdbcParamAdd\(\)](#) definierten Datentyp überein, gibt die Anweisung [ErrType](#) zurück.

Wird als Wert (var2) NULL übergeben, wird der Inhalt auf einen nicht definierten Wert gesetzt.
Die Spalte muss auf NULL gesetzt werden können.

-  Wird ein Wert vom Typ handle erwartet, kann nicht NULL übergeben werden.
Andernfalls wird ErrType zurückgeliefert.
Wurde bei OdbcParamAdd() der Datentyp TypeHandle definiert, muss im Parameterwert
(var2) der Deskriptor eines Memory-Objektes übergeben werden, ansonsten kommt der
Laufzeitfehler ErrHdlInvalid.

-  Die Variable des Memory-Objektes muss mit dem Datentyp handle deklariert sein.

Beispiel:

```
// Abfrage vorbereitet OdbcStm # tOdbcCon->OdbcPrepare('INSERT INTO Customer (ID,NAME,DESCRIPTION
```

Mögliche Laufzeitfehler:

Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.

Bei OdbcParamAdd() der Datentyp TypeHandle definiert und der Parameterwert (var2) ist kein Deskriptor eines Memory-Objektes.

ErrValueInvalid In (int1) ist kleiner als eins oder es gibt keinen Parameter mit dem angegebenen Index.

Kontakt

obj -> OdbcParamAdd(int1[, int2[, logic3]]) : int



Parameter für ODBC-Statement definieren

Deskriptor eines

obj ODBC-Statement-Objekts

int1 Datentyp

maximale Länge der

int2 Zeichenkette (optional)

Länge der Zeichenkette

logic3 ist variabel (optional)

Resultat int Fehlerwert

Verwandte Befehle,

Siehe OdbcPrepare(),

OdbcParamSet()

SQL-Statements können zum Zweck der einfacheren Handhabung parametrisiert werden. Dazu werden im SQL-Statement Fragezeichen anstelle der Parameter angegeben.

INSERT INTO Customer (ID,NAME) VALUES (1000,'vectorsoft AG')

Diese Anweisung fügt den Datensatz mit der ID 1000 in die mit ODBC verbundene Datenquelle ein. In der Regel liegen die Daten (hier ID und Name) jedoch nicht fix vor, sondern werden z. B. aus einer Eingabemaske eines Anwenders übernommen. Zu diesem Zweck könnte das Statement wie folgt angepasst werden:

INSERT INTO Customer (ID,NAME) VALUES (?,?)

Die durch ? gekennzeichneten Werte werden extern bezogen. Das Beispiel wird in CONZEPT 16 wie folgt realisiert:

```
tOdbcStm # tOdbcCon->OdbcPrepare('INSERT INTO Customer (ID,NAME) VALUES (?,?)');if (tOdbcStm > 0)
```

Durch die OdbcParamAdd()-Anweisungen werden dem Befehl zwei (für jedes Fragezeichen im SQL-Statement) Parameter mit dem entsprechenden Datentyp der Spalte hinzugefügt. Als (obj) wird der Deskriptor des OdbcStm-Objekts angegeben. In (int1) wird der Datentyp mit einer _Type...-Konstanten angegeben.



Bei den Datentypen TypeAlpha und TypeHandle ist der Parameter (int2) notwendig.
Er gibt die maximale Länge der Zeichenkette an.

Im Parameter (logic3) kann beim Datentyp TypeAlpha definiert werden, ob die in (int2) angegebene Länge eine variable Länge ist (true) oder eine Maximallänge (false).



Bei Angabe einer Maximallänge füllen einige ODBC-Treiber (bzw. das zugrunde liegende Datenbank-System) den Feldinhalt bis zur maximalen Feldlänge mit Leerzeichen auf.

Der Datentyp TypeHandle kann angegeben werden, um mit OdbcParamSet() ein Memory-Objekt an das ODBC-Statement zu übergeben. In diesem Fall werden die enthaltenen Daten des Memory-Objektes als SQL_LONGVARCHAR übertragen. Bei der Übertragung findet keine Zeichensatzkonvertierung statt.

Kontakt

Nach der Definition der Parameter, können die eigentlichen Werte mit dem Befehl OdbcParamSet() angegeben werden:

```
tOdbcStm->OdbcParamSet(1, 1000); tOdbcStm->OdbcParamSet(2, 'vectorsoft AG');
```

Bei der Ausführung des Statements mit OdbcExecute() werden die ? durch die aktuell definierten Werte ersetzt.

Die Anweisung kann folgende Werte zurückgeben:

- ErrOk

Die Anweisung wurde erfolgreich ausgeführt.

- ErrOdbcError

Bei der Durchführung ist ein Fehler aufgetreten. Genaue Informationen über die Fehlerursache können über die Fehler-Eigenschaften OdbcErr... des ODBC-Statement-Objekts abgefragt werden.

- ErrOutOfMemory

Zur Ausführung der Anweisung steht kein ausreichender Hauptspeicher zur Verfügung.

Mögliche Laufzeitfehler:

ErrHdIInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.

Kontakt

obj -> OdbcExecute() : int;  Vorbereitetes ODBC-

Statement ausführen

obj Deskriptor eines

ODBC-Statement-Objekts

Resultat **int** Fehlerwert 

Verwandte Befehle,
OdbcPrepare(),

Siehe

OdbcParamSet(),
OdbcExecuteDirect()

Ein mit OdbcPrepare() erfolgreich vorbereitetes SQL-Statement wird mit diesem Befehl durchgeführt. Der Befehl liefert bei erfolgreicher Durchführung ErrOk, sonst ErrOdbcError zurück. Genauere Informationen über die Fehlerursache können über die Fehler-Eigenschaften OdbcErr... des ODBC-Statement-Objekts abgefragt werden.

Nach der Ausführung der Anweisung kann die Ergebnismenge mit OdbcFetch() ausgelesen werden.

Ein vorbereitetes ODBC-Statement kann mehrfach hintereinander durchgeführt werden, ohne dass es einer erneuten Vorbereitung bedarf. Wurde ein ODBC-Statement mit Parametern versehen (siehe OdbcParamAdd()), können vor der Durchführung neue Parameter mit der Anweisung OdbcParamSet() gesetzt werden.

Mögliche Laufzeitfehler:

ErrHdIInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.

Kontakt

obj -> OdbcExecuteDirect(alpha1) : handle <img alt="Icon representing a connection object" data

Statement vorbereiten und ausführen

obj Deskriptor eines ODBC-Verbindungs-Objekts

alpha1 SQL-Statement

Resultat handle Deskriptor eines ODBC-Statement-Objekts



Verwandte Befehle, OdbcConnect()

Siehe [Odka-Punam](#) & [Odka-Ewan](#)

[OdbcPrepare\(\)](#), [OdbcExecute\(\)](#), [OdbcClose\(\)](#)

Mit dieser Anweisung wird ein SQL-Statement vorbereitet und sofort ausgeführt. Der Befehl liefert bei erfolgreicher Durchführung einen Deskriptor auf ein OdbcStm-Objekt, sonst ErrOdbcError zurück. Genauere Informationen über die Fehlerursache können über die Fehler-Eigenschaften OdbcErr... des ODBC-Verbindungs-Objekts abgefragt werden. Ist schon die Erstellung des ODBC-Statement-Objekts fehlgeschlagen, gibt die Anweisung ErrOdbcFunctionFailed zurück.

Nach der Ausführung der Anweisung kann die Ergebnismenge mit OdbcFetch() ausgelesen werden.

Das ODBC-Statement-Objekt steht zur Verfügung, bis die Verbindung zur Datenquelle getrennt oder das Objekt mit [OdbcClose\(\)](#) geschlossen wird.

Mögliche Laufzeitfehler:

ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcCon-Objekt übergeben.

Kontakt

obj -> OdbcClmInfo(int1) : int  [Informationen](#)
zu einer Spalte ermitteln
Deskriptor eines
obj ODBC-Statement-Objekts
int1 Nummer der Spalte
Resultat int Fehlerwert
[Verwandte Befehle](#),
Siehe [OdbcExecute\(\)](#),
[OdbcExecuteDirect\(\)](#)

Die Anweisung liefert Informationen zu einer Spalte, deren Index mit (int1) angegeben werden muss.
In (obj) muss ein Deskriptor auf ein OdbcStm-Objekt übergeben werden. Damit der Befehl erfolgreich durchgeführt werden kann, muss zuvor die Anweisung OdbcExecute() oder OdbcExecuteDirect() erfolgreich aufgerufen worden sein.

Im Gegensatz zu der Anweisung OdbcCatalogClm() können hier nur die Spalten der Ergebnismenge und nicht der gesamten Tabelle oder Datenquelle abgefragt werden.
Die zulässige Werte für (int1) liegen im Bereich von 1 bis zu dem Wert, den die Eigenschaft OdbcResCountClm des OdbcStm-Objekts liefert.

Nach der Durchführung der Anweisung sind folgende Eigenschaften im übergebenen Objekt gesetzt:

- OdbcClmName
- OdbcClmDataType
- OdbcClmSize
- OdbcClmDecimalDigits
- OdbcClmNullable

Der Befehl liefert bei erfolgreicher Durchführung ErrOk, sonst ErrOdbcError zurück.
Genauere Informationen über die Fehlerursache können über die Fehler-Eigenschaften OdbcErr... des ODBC-Statement-Objekts abgefragt werden.

Mögliche Laufzeitfehler:

Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.

Kontakt

obj -> OdbcClmData(int1, var2) : int  Wert einer Spalte aus Ergebnismenge lesen
Deskriptor auf ein obj ODBC-Statement-Objekt
int1 Nummer der Spalte Variable zur Aufnahme
var2 des Werts
Resultat int Fehlerwert
Verwandte Befehle,
OdbcExecute(),
Siehe OdbcExecuteDirect(),
OdbcClmInfo()

Die Anweisung liefert den Wert zu einer Spalte, deren Index mit (int1) angegeben werden muss. In (obj) muss ein Deskriptor auf ein OdbcStm-Objekt übergeben werden. Damit der Befehl erfolgreich durchgeführt werden kann, muss zuvor die Anweisung OdbcExecute() oder OdbcExecuteDirect() erfolgreich aufgerufen worden sein.

In (int1) wird der Index der Spalte in der Ergebnismenge angegeben. D. h. die zulässige Werte für (int1) liegen im Bereich von 1 bis zu dem Wert, den die Eigenschaft OdbcResCountClm des OdbcStm-Objekts liefert.

In (var2) sollte eine Variable angegeben werden, die einen kompatiblen Typ zu der Spalte in der Ergebnismenge besitzt. Der Typ der Spalte kann mit Hilfe der Anweisung OdbcClmInfo() ermittelt werden. Alternativ kann auch eine Variable vom Typ alpha übergeben werden. In diesem Fall findet die Konvertierung automatisch statt.

Wird ein Memory-Objekt angegeben, können Zeichenketten abgerufen werden, die länger als 8192 Zeichen sind. Die Daten werden immer an die aktuelle Position (Len) des Memory-Objektes angehängt. Bei der Übertragung findet keine Zeichensatzkonvertierung statt.

Der Befehl liefert bei erfolgreicher Durchführung ErrOk, sonst ErrOdbcError zurück. Genauere Informationen über die Fehlerursache können über die Fehler-Eigenschaften OdbcErr... des ODBC-Statement-Objekts abgefragt werden.

Mögliche Laufzeitfehler:

- ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.
ErrValueRange Die angegebene Variable (var2) ist vom Typ handle, enthält aber keinen Deskriptor auf ein Memory-Objekt.
Die zu lesenden Daten sind größer als der vom Memory-Objekt bereitgestellte Speicher. Dieses Fehler tritt auch bei der Option MemAutoSize auf, wenn kein Speicher mehr angefordert werden kann.
ErrMemExhausted Speicheranforderung fehlgeschlagen.

Kontakt

obj -> OdbcFetch([int1]) : int       Zeile

der Ergebnismenge lesen

obj Deskriptor auf ein
ODBC-Statement-,
-Tabellen- oder
-Spalten-Objekt
reserviert, muss 0

int1 sein

Resultat int Fehlerwert 

Verwandte Befehle,

OdbcExecute(),

Siehe OdbcExecuteDirect(),

OdbcCatalogTbl(),

OdbcCatalogClm()

Die Anweisung positioniert auf die nächste Zeile der Ergebnismenge. Wurde in der Ergebnismenge noch nicht gelesen, wird die erste Zeile gelesen. Die Ergebnismenge kann durch die Anweisungen OdbcCatalogTbl(), OdbcCatalogClm(), OdbcExecute() oder OdbcExecuteDirect() erzeugt worden sein. Der Deskriptor des entsprechenden Objekts muss als (obj) übergeben werden.

Der optionale Parameter (int1) wird für zukünftige Erweiterungen benötigt und sollte nicht verwendet werden. Wird ein Wert angegeben, muss er 0 sein.

Die Anweisung kann folgende Werte zurückgeben:

- ErrOk

Die Anweisung wurde erfolgreich durchgeführt. Die entsprechenden Informationen stehen in den Eigenschaften des übergebenen Objekts zur Verfügung. Soll die Ergebnismenge nach der Durchführung eines ODBC-Statements gelesen werden, können Informationen über die Spalten mit der Anweisung OdbcClmInfo() und die Werte mit OdbcClmData() ausgelesen werden.

- ErrOdbcNoData

In der Ergebnismenge konnte nicht auf die nächste oder erste Zeile positioniert werden. Es liegen keine (weiteren) Daten vor.

- ErrOdbcError

Beim Positionieren auf eine Zeile der Ergebnismenge ist es zu einem Fehler gekommen. Weitere Informationen zu dem Fehler können in den OdbcErr...-Eigenschaften des übergebenen Objekts ermittelt werden.

Beispiel:

```
// Abfrage durchführt OdbcStm # tOdbcCon->OdbcExecuteDirect('INSERT INTO Customer (ID,NAME) VALU
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein zulässiges Objekt

Kontakt

übergeben.

Kontakt

FldAttributes(int1, int2, int3[,int4]) :



int

ODBC-Attribute setzen/ermitteln

int1 Dateinummer

int2 Teildatensatznummer

int3 Feldnummer

 Neue Feldattribute (optional)

KeyFldAttrUpperCase Groß-/Kleinschreibung

 beachten

int4 KeyFldAttrUmlaut Umlaute beachten

KeyFldAttrSpecialChars Sonderzeichen nicht

 beachten

 0 Parameter löschen

Resultat int Aktuelle Feldattribute

Verwandte Befehle, ODBC-Schnittstelle,

Siehe

ODBC-Befehle



Dieser Befehl wird in der aktuellen Version 5.7 noch nicht unterstützt.

Beim Zugriff eines anderen Programms auf eine CONZEPT 16-Datenbank über die ODBC-Schnittstelle können unabhängig von den in der Datenbank vorliegenden Schlüsseln, Sortierfelder angegeben werden. Zum Beispiel in einem SELECT-Statement mit der Klausel ORDER BY.

Um die Sortierreihenfolge zu beeinflussen muss beim Einrichten der Datenquelle eine Startprozedur angegeben werden, in der mit dem Befehl FldAttributes() die Sortierparameter gesetzt werden. Sortierparameter können mit (int4 = 0) gelöscht werden.

Beispiele:

Die Datensätze sollen nach einem Suchwort sortiert ausgegeben werden. Das entsprechende ODBC-Statement lautet:

```
SELECT SWT_aSuchwort FROM SWT_D_Suchwort ORDER BY SWT_aSuchwort
```

Ohne die Verwendung der Anweisung FldAttributes() werden die Suchworte in folgender Reihenfolge zurückgegeben:

Aenderung

Zwischensumme

angeln

zeigen

Änderung

ändern

Die Reihenfolge entspricht der Wertigkeit der Zeichen in der ASCII-Tabelle.

Wird in der Startprozedur für das Feld SWT.aSuchwort die Anweisung FldAttributes(..., _KeyFldAttrUpperCase) angegeben, verändert sich die Reihenfolge

Kontakt

unter Verwendung des gleiche ODBC-Statements wie folgt:

Aenderung

angeln

zeigen

Zwischensumme

ändern

Änderung

Durch die Anweisung FldAttributes(..., _KeyFldAttrUmlaut) werden auch die Umlaute in alphabetischer Reihenfolge angezeigt:

ändern

Aenderung

Änderung

angeln

zeigen

Zwischensumme

Kontakt

ProcAdvertise(alpha1, alpha2[,
logic3]) : logic



"Stored Procedure" bekannt geben
Prozedur- und

alpha1

Funktionsname

Funktionsparameter als

logic3 In/Out-Parameter
registrieren (optional)

Resultat logic Erfolg des Befehls

Verwandte Befehle,

Siehe ODBC-Schnittstelle -
Datenquellen einrichten

Diese Anweisung wirkt sich nur bei der Verwendung des ODBC-Treibers von CONZEPT 16 aus. Der Befehl kann in einer Startprozedur einer ODBC-Datenquelle (siehe [ODBC-Schnittstelle - Funktionen](#)) oder in einer "Stored Procedure" ausgeführt werden.

Mit dem Befehl wird die in (alpha1) übergebene Funktion der ODBC-Schnittstelle unter dem im (alpha2) angegebenem symbolischen Namen bekanntgegeben. Die Funktion kann dann als "Stored Procedure" von der ODBC-Applikation ausgeführt werden, insofern diese die "Stored Procedure"-Funktionalität unterstützt. Funktionsparameter können dabei wie gewohnt an die "Stored Procedure" übergeben werden.

In (logic3) wird optional mit true definiert, dass alle Funktionsparameter der bekanntgegebenen Funktion als In/Out-Parameter (Ähnlich wie bei einer Call-by-Reference) verwendet werden. Das bedeutet, dass sich Änderungen der Werte in der "Stored Procedure" auch auf die ursprünglich übergebenen Variablen auswirken. Ein Markieren der Funktionsparameter in CONZEPT 16 mit var entfällt dabei. Wird (logic3) nicht oder mit false angegeben, werden alle Parameter nur als In-Parameter verwendet (Notwendig falls Konstante Werte als Parameter übergeben werden sollen). Ein einzelnes festlegen der In/Out-Funktionalität der Parameter ist nicht möglich.

Konnte die Funktion bekannt gegeben werden, erfolgt die Rückgabe von true. Im anderen Fall wird false zurückgegeben. Funktionen können nicht bekanntgegeben werden, wenn sie Übergabeparameter erwarten, die nicht von ODBC unterstützt werden (Arrays und zusammengesetzte Datentypen) oder nicht definiert sind.



Innerhalb einer "Stored Procedure" können keine Oberflächen-Befehle verarbeitet werden.

Beispiele:

Mit der folgenden Anweisung wird die Funktion Start in der Prozedur ODBC bekanntgegeben. Die Parameter werden dabei nur als In-Parameter registriert. In der Applikation kann diese Funktion dann mit dem Namen InterfaceInit gestartet werden:

```
ProcAdvertise('ODBC:Start', 'InterfaceInit', false);
```

Kontakt

Innerhalb der Applikation kann diese Funktion dann über einen entsprechenden Menüpunkt oder ähnliches aufgerufen werden. Der Aufruf kann ebenfalls innerhalb eines SQL-Statements erfolgen:

```
{ CALL InterfaceInit() }
```

Sind bei der Funktion Übergabeparameter definiert, müssen diese ebenfalls beim Aufruf angegeben werden:

```
{ CALL InterfaceInit( 1, 3.5, 'Start' ) }
```

Socket-Befehle

Befehle für den Datenaustausch über eine TCP/IP-Socket-Verbindung Verwandte

Befehle, Liste

sortiert nach

Siehe Gruppen,

Alphabetische

Liste aller

Befehle

Befehle

- SckClose
- SckConnect
- SckInfo
- SckListen
- SckRead
- SckReadMem
- SckStartTls
- SckWrite
- SckWriteMem

Konstanten

- SckAddrLocal
- SckAddrPeer
- SckBuffered
- SckCertificateCN
- SckDefaultKeepAlive
- SckHostName
- SckKeepAlive
- SckLine
- SckNoTLSv1
- SckOptBind
- SckOptDelay
- SckOptDontLinger
- SckOptVerify
- SckPortLocal
- SckPortPeer
- SckProxySOCKSv4
- SckProxySOCKSv4a
- SckProxySOCKSv5
- SckReadMax
- SckReadyRead
- SckTimeout
- SckTlsHigh
- SckTlsLow
- SckTlsMax
- SckTlsMed
- SckTlsSNI
- SckVersionTLS
- SckVol

Kontakt

- SckVolRead
- SckVolWrite

Kontakt

obj -> SckClose() |  | Socket-

Verbindung beenden

obj Socket-Deskriptor

Verwandte

Siehe Befehle,

SckConnect()

Mit dieser Funktion wird eine mittels SckConnect() geöffnete Verbindung wieder geschlossen. In (obj) wird der Socket-Deskriptor übergeben.

Sofern der Deskriptor (obj) nicht definiert ist oder keinen Socket bezeichnet, erfolgt ein Laufzeitfehler (ErrHdIInvalid).



In der Beispiel-Datenbank "CodeLibrary" befindet sich ein Beispiel zur Kommunikation mit einem FTP-Server.

Mögliche Laufzeitfehler:

ErrHdIInvalid Der angegebene Deskriptor ist ungültig oder verweist nicht auf eine Socket-Verbindung.

Kontakt

`SckConnect(alpha1, word2[,
int3[, int4[, alpha5[, word6[,
alpha7[, alpha8]]]]]]) : handle`
Socket-Verbindung aufbauen



alpha1 Host

word2 Portnummer

Optionen (optional)
[SckOptDelay](#)

Schaltet den

[SckOptDontLinger](#)

Nagle-Algorithmus ein
Schaltet den

[SckProxySOCKSv4](#)

Linger-Modus aus
SOCKS-Proxyserver

[SckProxySOCKSv4a](#) SOCKS-Proxyserver

Version 4
Version 4a
SOCKS-Proxyserver

[SckProxySOCKSv5](#)

Version 5
Verschlüsselung mit
maximalen

[SckTlsMax](#)

Sicherheitsanforderungen
Verschlüsselung mit
hohen

int3

[SckTlsMed](#)

Sicherheitsanforderungen
Verschlüsselung mit
durchschnittlichen

[SckTlsLow](#)

Sicherheitsanforderungen
Verschlüsselung mit
niedrigen

[SckNoTLSv1](#)

Sicherheitsanforderungen
Verschlüsselung ohne

[SckTlsSNI](#)

Unterstützung von TLS 1.x
Verwendung von Server
Name Indication (SNI)
beim verschlüsselten

[SckOptVerify](#)

Verbindungsaufbau
Überprüfung des
serverseitigen Zertifikats

[SckOptBind](#)

Verbindung an bestimmte
IP-Adresse und / oder
bestimmten Port binden

int4 Timeout in Millisekunden (optional)

alpha5 Proxy-Server / Bind-Adresse
(optional)
Proxy-Portnummer /

word6 Bind-Portnummer (optional)

alpha7 Proxy-Benutzer (optional)

alpha8 Proxy-Password (optional)

Kontakt

Resultat **handle** Deskriptor / Fehlerwert 
Verwandte Befehle, SckClose(),

Siehe

[SckStartTls\(\)](#)

Mit dieser Anweisung wird eine TCP/IP Verbindung zum Host (alpha1) auf Port (word2) aufgebaut. Der Host kann entweder als IP-Adresse (a.b.c.d) oder als Name angegeben werden. Bei der Verwendung des Namens ist zu beachten, dass dieser auch auf eine IP-Adresse umgesetzt werden kann (normalerweise durch einen DNS-Server). Durch die Verwendung eines Präfixes vor dem Namen kann die Auswahl der IP-Adresse und somit des Protokolls gesteuert werden. Folgende Präfixe können angegeben werden:

ip4: ausschließlich IPv4

ip4f: bevorzugt IPv4, IPv6 wenn es keine IPv4-Adresse gibt

ip6: ausschließlich IPv6

ip6f: bevorzugt IPv6, IPv4 wenn es keine IPv6-Adresse gibt

Wird kein Präfix angegeben, wird automatisch ip4f: verwendet.

Beispiel:

```
SckConnect('ip6f:www.vectorsoft.de', 80);
```



In der Beispiel-Datenbank "CodeLibrary" befindet sich ein Beispiel zur Kommunikation mit einem FTP-Server.

Durch die Portnummer wird der Service des Socket-Servers spezifiziert. Für viele Protokolle sind bestimmte Portnummern standardisiert:

FTP : 21

TELNET : 23

SMTP : 25

HTTP : 80

HTTPS : 443

POP3 : 110

Die Option SckOptDelay schaltet den Nagle-Algorithmus des TCP/IP-Protokoll-Stacks ein. Damit werden beim Versenden nach Möglichkeit mehrere Datenpakete zu einem größeren Paket zusammengefasst. Dies ist standardmäßig abgeschaltet, da die Verwendung zu einer deutlich höheren Latenz führen kann.

Da die Funktion SckWrite() auch einen gepufferten Modus anbietet, braucht

SckOptDelay nur in Spezialfällen verwendet werden.

Die Option SckOptDontLinger schaltet den Linger-Modus des TCP/IP-Protokoll-Stacks aus. Dadurch wird bei einem SckClose() nicht mehr auf das Versenden noch eventuell ungesendeter Daten gewartet.

Durch die Optionen SckTlsMax, SckTlsHigh, SckTlsMed und SckTlsLow kann eine verschlüsselte Socket-Verbindung unter Verwendung der Transport Layer Security aufgebaut werden. Basierend auf dieser Sicherungsschicht kann beispielsweise das HTTPS-Protokoll verwendet werden. Der Verbindungsauflauf erfolgt mit der Angabe

Kontakt

von einer dieser Konstanten. Der Aufruf von

`SckConnect(<Server>, <Port>, _SckTlsMed)` stellt eine verschlüsselte Verbindung zum Server unter Verwendung von SSL 3.0 oder TLS 1.x her, eine unverschlüsselte Verbindung oder die Verwendung von SSL 2.0-Verschlüsselungen wird abgelehnt.



Eine Verschlüsselung kann auch nachträglich mit `SckStartTls()` initiiert werden.

Falls der Server keine akzeptierte Version oder überhaupt keine Verschlüsselung anbietet, liefert `SckConnect()` den Fehler `ErrSckTlsConnect` zurück. Der Fehlerwert kommt auch dann, wenn das TLS-Handshake mit dem Server fehlschlägt. Eine Überprüfung der Gültigkeit des Server-Zertifikats ist derzeit nicht möglich.

Falls der Server Probleme mit den TLS 1.x-Protokollen hat, kann die Option `SckNoTLSv1` mit `SckTlsMed` oder `SckTlsLow` kombiniert werden, um die Unterstützung auf das Protokoll SSL 3.0 zu beschränken.

Die verwendete TLS-Version kann mit `SckInfo(..., SckVersionTLS)` abgefragt werden.

Soll das serverseitig verwendete Zertifikat überprüft werden, muss die Angabe in (int3) mit `SckOptVerify` kombiniert werden. Voraussetzung für eine erfolgreiche Prüfung von Zertifikaten ist eine Sammlung von Stammzertifikaten vertrauenswürdiger Organisationen (root CAs). Diese Sammlung wird als Datei common\ca-bundle.crt im CONZEPT 16-Datenverzeichnis abgelegt. Zur vollständigen Überprüfung muss zusätzlich der "Common Name" in dem Zertifikat mit dem Server-Namen verglichen werden. Der "CN" kann mit der Anweisung `SckInfo(..., SckCertificateCN)` ermittelt werden.

Tritt bei der Überprüfung des Zertifikats ein Fehler auf, liefert der Befehl `ErrSckTlsCertificateVerify` zurück. Der Fehlerwert der Überprüfung befindet sich in der Eigenschaft `CertificateError` des `System`-Objekts.

Zusätzlich zu den Konstanten `SckTlsMax`, `SckTlsHigh`, `SckTlsMed` und `SckTlsLow` kann mit `SckTlsSNI` beim Verbindungsauflauf die Server Name Indication (SNI) eingesetzt werden. Wird eine `SckProxy...-Option` angegeben, wird der Name des Proxy-Servers (alpha5) verwendet, andernfalls der Hostname (alpha1). Soll sich bei Verwendung eines Proxys SNI auf den normalen Host beziehen, muss `SckStartTls()` verwendet werden.

Nach dem Aufbau der Verbindung kann diese genauso verwendet werden wie eine unverschlüsselte Verbindung.

Verschlüsselte Socket-Verbindungen können aktuell in folgenden Umgebungen verwendet werden:

- Standard- und Advanced-Client
- SOA-Service
- Programmierschnittstelle
- Druck-Prozessor

Optional kann in (int4) ein Timeout definiert werden. Dieser Wert (in Millisekunden) wird beim Öffnen, Lesen und Schreiben des Sockets verwendet. Nach Ablauf des

Kontakt

Timeouts wird entweder der Fehlerwert ErrSckConnectFailed (beim Öffnen) oder ErrTimeout (beim Lesen und Schreiben) zurückgeliefert. Standardmäßig wird ein Timeout von 5000 ms verwendet.

Durch die _SckProxy...-Optionen kann ein SOCKS-Proxyserver verwendet werden, wobei die Versionen 4, 4a und 5 (ohne Authentifizierung) unterstützt werden. Bei Benutzung eines Proxys muss in (alpha5) der Name oder die IP-Adresse des Proxyservers angegeben werden. Der Port des Proxyservers kann optional in (word6) übergeben werden, standardmäßig wird 1080 verwendet. Bei Benutzung von SOCKS v4 (ohne a) ist zu beachten, dass der Host in (alpha1) nur als IP-Adresse angegeben werden kann, da erst ab Version 4a eine Namensauflösung möglich ist.

Benötigt der Proxyserver eine Authentifizierung, kann ein Benutzer und sein Passwort in (alpha7) und (alpha8) angegeben werden.

Mit der Option SckOptBind wird die Verbindung an die IP-Adresse (alpha5) und / oder den Port (word6) gebunden. Dies ist beispielsweise dann sinnvoll, wenn ein Computer mehrere IP-Adressen hat und bei der Verbindung zu einem anderen Rechner eine bestimmte IP-Adresse, z. B. durch die Firewall, vorausgesetzt wird. Ist kein Port (word6) angegeben, wird automatisch ein freier Port gewählt. Ist ein angegebener Port bereits belegt, wird als Resultat ErrSckBindFailed zurückgegeben. Kann mit der Quelladresse keine Verbindung zur Zieladresse (alpha1) aufgebaut werden, kommt als Resultat ErrSckConnectFailed. Dies ist beispielsweise der Fall, wenn sich Quell- und Zieladresse in anderen Subnetzen befinden.

Mögliche Fehlerwerte:

ErrSckNoLib

Das TCP/IP-Protokoll konnte nicht initialisiert werden

ErrSckHostUnknown

Der Name des Ziel-Hosts konnte nicht in eine IP-Adresse übersetzt werden

ErrSckCreate

Der Socket konnte nicht angelegt werden

ErrSckConnectFailed

Es konnte keine Verbindung zum Ziel-Host aufgebaut werden

ErrSckBindFailed

Es konnte keine Zuordnung zum angegebenen Port erfolgen

ErrSckProxyUnknown

Der Name des Proxys konnte nicht in eine IP-Adresse übersetzt werden

ErrSckProxyRefused

Der Proxy hat den Verbindungswunsch abgelehnt

ErrSckProxyConnectFailed

Es konnte keine Verbindung zum Proxy aufgebaut werden

ErrSckProxyRead

Bei der Kommunikation mit dem Proxy ist ein Lesefehler aufgetreten

ErrSckProxyWrite

Bei der Kommunikation mit dem Proxy ist ein Schreibfehler aufgetreten

ErrSckTlsConnect

Der verbundene Server unterstützt keine der angegebenen verschlüsselten Verbindungen oder der TLS-Handshake ist fehlgeschlagen.

ErrSckTlsCertificateVerify Fehler bei der Überprüfung des serverseitigen Zertifikats

Kontakt

obj -> SckInfo(int1[, int2]) : alpha                            <img alt="Icon of a network connection" data-bbox

Kontakt

Ähnlichkeitsvergleich durchgeführt werden (siehe SckCertificateCN).

Wird in (int1) SckReadyRead angegeben, muss in (int2) eine Wartezeit in Millisekunden angegeben werden. Sind bis zu diesem Zeitpunkt keine Daten vorhanden, wird ErrTimeout zurückgegeben. Sind Daten vorhanden wird '0' zurückgegeben.

Sofern der Deskriptor (obj) nicht definiert ist oder keinen Socket bezeichnet, erfolgt der Laufzeitfehler (ErrHdlInvalid).

Kontakt

SckListen(alpha1, word2[, int3[, handle4]]) : handle



Passive Socket-Verbindung aufbauen

alpha1 Host

word2 Portnummer

Optionen (optional)

SckOptDelay

Schaltet den
Nagle-Algorithmus

int3

ein

SckOptDontLinger Schaltet den

Linger-Modus aus

handle4 Deskriptor des Frame-Objektes

(optional)

Resultat handle Deskriptor / Fehlerwert

Verwandte Befehle, SckConnect(),

Siehe

EvtSocket

Mit dem Befehl SckListen() wird ein Socket auf dem Port (word2) geöffnet, der anschließend für eingehende Verbindungen zur Verfügung steht.

In (int3) können die Optionen SckOptDontLinger und SckOptDelay angegeben werden. In (alpha1) wird normalerweise ein Leerstring übergeben, wodurch der Socket an alle IP-Adressen des Rechners gebunden wird. Sofern der Socket nur an eine IP-Adresse gebunden werden soll, muss diese in (alpha1) angegeben werden.

Beim Öffnen einer neuen Verbindung wird ein EvtSocket generiert. In (int4) kann der Deskriptor des Frame-Objekts angegeben werden, welches das Ereignis erhält. Wird (handle4) nicht angegeben oder auf 0 gesetzt, erhalten alle Top-Level-Frames (die Frames ohne Eltern-Objekt) das Ereignis. Eine entsprechende Prozedurfunktion für das Ereignis EvtSocket muss beim jeweiligen Fenster angegeben werden.

Das Timeout einer Verbindung kann erst dann erfolgen, wenn eine Verbindung aufgebaut wurde. Das Timeout kann dann mit dem Befehl SckInfo(..., SckTimeout, ...) gesetzt werden.

Solange der mit SckListen() erzeugte Socket offen ist, können beliebig oft neue Verbindungen über diesen Port hergestellt werden. Bei jedem Verbindungsaufbau wird dabei ein Ereignis erzeugt. Der Socket wird mit SckClose() wieder geschlossen. Der Deskriptortyp des Sockets ist HdlSocketListen, andere Operationen als SckClose() sind auf diesem Sockettyp nicht möglich.

Kontakt

obj -> SckRead(int1,
var2[, int3]) : int
Daten vom Socket lesen
obj Socket-Deskriptor
Optionen
 SckLine komplette
 Zeile lesen
 SckReadMax Lesen bis
int1 maximale
 Länge / Lesen
 aller
 empfangener
 Zeichen
var2 Feld, Variable oder Array
 Anzahl zu lesender Bytes
int3 (optional)



Resultat int Anzahl der gelesenen
 Bytes oder Fehlerwert
Verwandte Befehle,
Siehe SckWrite()

Mit dieser Funktion werden Daten vom Socket (obj) gelesen. Durch die Angabe einer Option in (int1) kann die Anzahl der Zeichen auf eine Zeile oder auf die Größe der in (var2) angegebenen Variablen begrenzt werden.



In der Beispiel-Datenbank "CodeLibrary" befindet sich ein Beispiel zur Kommunikation mit einem FTP-Server.

In (var2) muss ein Datenbankfeld, eine Variable oder ein Array angegeben werden. Arrays aus Alphafeldern sind dabei nicht zulässig. Sofern (int3) nicht angegeben ist, wird die der Größe der Variablen (var2) entsprechende Anzahl von Bytes oder eine Zeile eingelesen. Der Wert in (int3) kann daher auch nicht größer als die Größe der 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.

Je nach Datentyp werden folgende Formate benutzt:

- alpha

Es kann sowohl eine feste als auch eine variable Anzahl von Zeichen eingelesen werden.

Bei einer festen Anzahl steht in (int3) die Anzahl der zu lesenden Bytes. Bei einer variablen Anzahl wird (int3) weggelassen und die Option SckReadMax in (int1) verwendet. In diesem Fall werden soviele Zeichen eingelesen, wie momentan empfangen wurden oder bis die maximale Länge erreicht ist.

Alternativ kann mit der Option SckLine eine komplette Zeile gelesen werden, die durch CR/LF abgeschlossen ist. CR/LF wird dabei nicht in die Variable übertragen. Diese Option wird bei vielen zeilenorientierten Protokollen benutzt

Kontakt

(SMTP, POP3 usw.).

- **float**

Es werden **8 Bytes** gelesen. Das Format entspricht dem IEEE-Double.

- **word**

Es werden **2 Bytes** gelesen. Das Format entspricht dem Intel-Wortformat (16 Bit - little endian).

- **int**

Es werden **4 Bytes** gelesen. Das Format entspricht dem Intel-Doppelwortformat (32 Bit - little endian).

- **date**

Es werden **4 Bytes** gelesen. (Byte 1 = Tag, Byte 2 = Monat, Byte 3 = Jahr, Byte 4 ist grundsätzlich leer.)

- **time**

Es werden **4 Bytes** gelesen. (Byte 1 = Stunde, Byte 2 = Minute, Byte 3 = Sekunde, Byte 4 = Hundertstelsekunde.)

- **logic**

Es wird **1 Byte** gelesen (Wert gleich 0 entspricht false, Wert gleich 1 entspricht true).

Mögliche Fehlerwerte:

<u>ErrHdlInvalid</u>	(obj) ist nicht definiert oder kein Socket-Deskriptor.
<u>ErrSckDown</u>	Der Socket wurde außerhalb der Applikation geschlossen
<u>ErrSckSelect</u>	Bei der Socketabfrage ist ein Fehler aufgetreten
<u>ErrSckRead</u>	Beim Lesen des Sockets ist ein Fehler aufgetreten
<u>ErrSckReadOverflow</u>	empfangene Zeile länger als die angegebene Variable
	Beim Lesen oder Schreiben des Sockets ist ein Timeout
<u>ErrTimeout</u>	aufgetreten

Kontakt

obj -> SckReadMem(int1, handle2, int3,
int4) : int



Vom Socket in Memory-Objekt lesen

obj Socket-Deskriptor

Optionen

SckLine komplette Zeile lesen

int1 SckReadMax Lesen bis maximale Länge /
 Lesen aller empfangener
 Zeichen

handle2 Deskriptor des Memory-Objekts

int3 Startposition im Memory-Objekt

int4 Anzahl der zu lesenden Bytes

Resultat int Anzahl der gelesenen Bytes oder
 Fehlerwert

Siehe Verwandte Befehle, SckWriteMem()

Mit dieser Funktion werden Daten vom Socket (obj) gelesen. Durch die Angabe einer Option in (int1) kann die Anzahl der Zeichen auf eine Zeile oder auf die Menge der verfügbaren Daten begrenzt werden. In (handle2) muss der Deskriptor eines Memory-Objekts angegeben werden. In (int3) wird die Startposition und (int4) die Anzahl der maximal zu lesenden Bytes übergeben. Gegebenenfalls wird der Wert der Eigenschaft Len erhöht.

Das Resultat gibt die Anzahl der gelesenen Bytes zurück. Ist das Resultat negativ ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder (handle2) übergebene Deskriptor ist ungültig.

ErrMemExhausted Der Speicher konnte nicht angefordert werden.

ErrValueRange Der in (int3) oder (int4) übergebene Wert ist außerhalb des zulässigen
 Bereichs.

Kontakt

obj -> SckStartTls(int1[, alpha2]) : int  TLS-Verschlüsselung für
Socket-Verbindung einleiten

obj Socket-Deskriptor

Optionen

SckTlsMax

Verschlüsselung mit
maximalen

SckTlsHigh

Sicherheitsanforderungen
Verschlüsselung mit
hohen

SckTlsMed

Sicherheitsanforderungen
Verschlüsselung mit
durchschnittlichen

int1

SckTlsLow

Sicherheitsanforderungen
Verschlüsselung mit
niedrigen

SckNoTLSv1

Verschlüsselung ohne
Sicherheitsanforderungen

SckTlsSNI

Unterstützung von TLS 1.x
Verwendung von Server
Name Indication (SNI)
beim verschlüsselten
Verbindungsauftbau

SckOptVerify

Überprüfung des
serverseitigen Zertifikats

Hostname bei Verwendung von SNI

alpha2 (optional)

Resultat int Fehlerwert 

Siehe Verwandte Befehle, SckConnect()

Mit dieser Funktion wird für eine mittels SckConnect() geöffnete Verbindung eine TLS-Verschlüsselung eingeleitet. In (obj) wird der Socket-Deskriptor übergeben.

Durch die Optionen (int1) SckTlsMax, SckTlsHigh, SckTlsMed und SckTlsLow wird der Sicherheitslevel der verschlüsselten Verbindung unter Verwendung der Transport Layer Security definiert. Der Aufruf von SckStartTls(Obj, _SckTlsMed) stellt eine verschlüsselte Verbindung zum Server unter Verwendung von SSL 3.0 oder TLS 1.x her, eine unverschlüsselte Verbindung oder die Verwendung von SSL 2.0-Verschlüsselungen wird abgelehnt.

Falls der Server keine akzeptierte Version oder überhaupt keine Verschlüsselung anbietet, liefert SckStartTls() den Fehler ErrSckTlsConnect zurück. Der Fehlerwert kommt auch dann, wenn das TLS-Handshake mit dem Server fehlschlägt. Eine Überprüfung der Gültigkeit des Server-Zertifikats ist derzeit nicht möglich.

Falls der Server Probleme mit den TLS 1.x-Protokollen hat, kann die Option SckNoTLSv1 mit SckTlsMed oder SckTlsLow kombiniert werden, um die Unterstützung auf das Protokoll SSL 3.0 zu beschränken.

Die verwendete TLS-Version kann mit SckInfo(..., SckVersionTLS) abgefragt werden.

Kontakt

Soll das serverseitig verwendete Zertifikat überprüft werden, muss die Angabe in (int3) mit SckOptVerify kombiniert werden. Voraussetzung für eine erfolgreiche Prüfung von Zertifikaten ist eine Sammlung von Stammzertifikaten vertrauenswürdiger Organisationen (root CAs). Diese Sammlung wird als Datei common\ca-bundle.crt im CONZEPT 16-Datenverzeichnis abgelegt. Zur vollständigen Überprüfung muss zusätzlich der "Common Name" in dem Zertifikat mit dem Server-Namen verglichen werden. Der "CN" kann mit der Anweisung SckInfo(..., SckCertificateCN) ermittelt werden.

Tritt bei der Überprüfung des Zertifikats ein Fehler auf, liefert der Befehl ErrSckTlsCertificateVerify zurück. Der Fehlerwert der Überprüfung befindet sich in der Eigenschaft CertificateError des System-Objekts.

Zusätzlich zu den Konstanten SckTlsMax, SckTlsHigh, SckTlsMed und SckTlsLow kann mit SckTlsSNI beim Verbindungsaufbau die Server Name Indication (SNI) eingesetzt werden. Dafür muss der Name des Zielhosts im Argument (alpha2) angegeben werden.

Beispiel zur Kommunikation mit einem verschlüsseltem Server durch einen HTTP-Proxy

```
// Verschlüsselte Verbindung durch HTTP-Proxy aufbauensub ConnectProxy(    aProxyName
    // Antwort empfangen      tRsp # HttpOpen(_HttpRecvResponse,tSck);      if (tRsp > 0)      {
// HTTP-Header im Debugger ausgebensub DbgHeader(    aRsp                      : handle;)  local {
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der angegebene Deskriptor (obj) ist ungültig oder verweist nicht auf eine Socket-Verbindung.

Kontakt

obj -> SckWrite(int1, var2[,
int3]) : int



Daten auf Socket schreiben
obj Socket-Deskriptor

Optionen
SckBuffered Daten puffern

int1 SckLine nach
 alphanumerischen
 Wert automatisch
 ein CR/LF senden

var2 zu schreibender Wert
 Anzahl zu schreibender Bytes

int3 (optional)
 Anzahl der geschriebenen

Resultat int
 Bytes oder Fehlerwert

Siehe Verwandte Befehle, SckRead()

Mit dieser Funktion werden Daten auf den Socket (obj) geschrieben.



In der Beispiel-Datenbank "CodeLibrary" befindet sich ein Beispiel zur Kommunikation mit einem FTP-Server.

Mit der Option SckLine in (int1) wird nach einem alphanumerischen Wert automatisch ein CR/LF gesendet.

Mit der Option SckBuffered werden die Daten zunächst zwischengespeichert und entweder beim Erreichen der Puffergrenze (4 KB) oder durch den Aufruf von SckWrite() mit einer Länge von 0 (int3) geschrieben. Dadurch werden die Daten von mehreren SckWrite()-Aufrufen für das Versenden zusammengefasst.

In (var2) kann ein Datenbankfeld, eine Variable, ein Array oder eine Zeichenketten-Konstante angegeben werden. Arrays aus Alphafeldern sind dabei nicht zulässig. Sofern (int3) nicht angegeben ist, wird die der Größe der Variablen (var2) entsprechende Anzahl von Bytes geschrieben. Der Wert in (int3) kann daher auch nicht größer als die Größe der Variablen selbst sein.

Das Resultat gibt die Anzahl der geschriebenen Bytes zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.

Sofern der Deskriptor (obj) nicht definiert ist oder keinen Socket bezeichnet, erfolgt ein Laufzeitfehler (ErrHdlInvalid).

Mögliche Fehlerwerte:

ErrSckDown Der Socket wurde außerhalb der Applikation geschlossen

ErrSckSelect Bei der Socketabfrage ist ein Fehler aufgetreten

ErrSckWrite Beim Schreiben des Sockets ist ein Fehler aufgetreten

ErrTimeout Beim Lesen oder Schreiben des Sockets ist ein Timeout aufgetreten

Kontakt

obj -> SckWriteMem(int1, handle2, int3, int4) :



int

Vom Memory-Objekt auf den Socket schreiben

obj Socket-Deskriptor

 Optionen

SckBuffered Daten puffern

int1 SckLine nach den Daten automatisch ein

 CR/LF senden

handle2 Deskriptor eines Memory-Objekts

int3 Startposition im Memory-Objekt

int4 Anzahl der zu schreibenden Bytes

Resultat int Anzahl geschriebener Bytes oder Fehlerwert Siehe

Verwandte Befehle, SckReadMem()

Mit dieser Funktion werden Daten auf den Socket (obj) geschrieben. Mit der Option SckLine in (int1) wird nach den Daten automatisch ein CR/LF gesendet. Mit der Option SckBuffered werden die Daten zunächst zwischengespeichert und entweder beim Erreichen der Puffergrenze (4 KB) oder durch den Aufruf von SckWrite() mit einer Länge von 0 geschrieben. Dadurch werden die Daten von mehreren SckWriteMem()-Aufrufen für das Versenden zusammengefasst.

In (handle2) muss der Deskriptor eines Memory-Objekts angegeben werden. In (int3) wird die Startposition und (int4) die Anzahl der zu schreibenden Bytes übergeben.

Das Resultat gibt die Anzahl der geschriebenen Bytes zurück. Ist das Resultat negativ, ist ein Fehler aufgetreten und das Resultat enthält den Fehlerwert.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) oder (handle2) übergeben Deskriptor ist ungültig.

 Der in (int3) oder (int4) übergebene Wert ist außerhalb des

ErrValueRange zulässigen Bereichs.

Kontakt

Konstanten für Socket-Befehle

<u>Siehe Alle Befehle</u>	
<u>SckBuffered</u>	Daten puffern
<u>SckLine</u>	CR/LF verarbeiten
<u>SckReadMax</u>	Lesen bis maximale Länge / Lesen aller empfangener Zeichen
<u>SckAddrLocal</u>	Die lokale IP-Adresse
<u>SckAddrPeer</u>	Die IP-Adresse des Zielhosts
<u>SckCertificateCN</u>	"Common Name" des Zertifikats
<u>SckDefaultKeepAlive</u>	Keep-Alive-Zeit auf den Vorgabewert setzen
<u>SckHostName</u>	Der lokale Hostname
<u>SckPortLocal</u>	Der lokale IP-Port
<u>SckPortPeer</u>	Der IP-Port des Zielhosts
<u>SckVol</u>	Das übertragene Datenvolumen in Bytes
<u>SckVolRead</u>	Das empfangene Datenvolumen in Bytes
<u>SckVolWrite</u>	Das gesendete Datenvolumen in Bytes
<u>SckReadyRead</u>	Ermitteln ob Daten zum Lesen vorhanden sind
<u>SckOptDelay</u>	Schaltet den Nagle-Algorithmus ein
<u>SckOptDontLinger</u>	Schaltet den Linger-Modus aus
<u>SckProxySOCKSv4</u>	Version 4 des SOCKS-Proxyserver verwenden
<u>SckProxySOCKSv4a</u>	Version 4a des SOCKS-Proxyserver verwenden
<u>SckProxySOCKSv5</u>	Version 5 des SOCKS-Proxyserver verwenden

Kontakt

_SckAddrLocal

Lokale IP-Adresse

Wert 1

Siehe SckInfo()

Option bei SckInfo() - Die lokale IP-Adresse wird ermittelt.

Kontakt

_SckAddrPeer

IP-Adresse des Zielhosts

Wert 2

Siehe SckInfo()

Option bei SckInfo() - Die IP-Adresse des Zielhosts ermitteln.

Kontakt

_SckBuffered

Daten puffern

Wert 4 / 0x04

Siehe SckWrite()

Option bei SckWrite() - Es werden die Daten zunächst zwischengespeichert und entweder beim Erreichen der Puffergrenze (4 KB) oder durch den Aufruf von SckWrite() mit einer Länge von 0 (int4) geschrieben.

Dadurch werden die Daten von mehreren SckWrite()-Aufrufen für das Versenden zusammengefasst.

Kontakt

_SckCertificateCN

"Common Name" des Zertifikats ermitteln

Wert 13

Siehe SckInfo()

Option bei SckInfo() - Die CN-Komponente (Common Name) des Zertifikats wird ermittelt. Der Name kann anschließend mit dem Hostnamen verglichen werden, um zu verifizieren, ob das Zertifikat zu dem Server gehört, mit dem eine Verbindung aufgebaut wurde.

Beispiel:

```
tHost # 'www.vectorsoft.de' try{  tSck # SckConnect(tHost, 443, _SckTlsHigh | _SckOptVerify);  if
```

Kontakt

_SckDefaultKeepAlive

Keep-Alive-Zeit auf den Vorgabewert setzen

Wert -1

Siehe SckInfo()

Option bei SckInfo() - Die Keep-Alive-Zeit wird auf den eingestellten Vorgabewert (siehe SOA-Service - Konfigurationsdatei) gesetzt.

Kontakt

_SckHostName

Lokaler Hostname

Wert 0

Siehe SckInfo()

Option bei SckInfo() - Den lokalen Hostnamen ermitteln.

Kontakt

_SckKeepAlive

Keep-Alive der Verbindung setzen / abfragen

Wert 11

Siehe SckInfo()

Option bei SckInfo() - Mit dieser Option kann die Keep-Alive-Zeit abgefragt (zwei Parameter) oder gesetzt (drei Parameter) werden.

Die Keep-Alive-Zeit wird in Millisekunden angegeben.

Kontakt

_SckLine

CR/LF verarbeiten

Wert 2 / 0x02

Siehe SckRead(),

SckWrite()

Option bei SckRead() - Es wird eine komplette Zeile gelesen, die durch CR/LF abgeschlossen ist.

CR/LF wird dabei nicht in die Variable übertragen. Diese Option wird bei vielen zeilenorientierten Protokollen benutzt (SMTP, POP3 usw.).

Option bei SckWrite() - Es wird nach einem alphanumerischen Wert automatisch ein CR/LF gesendet.

Kontakt

_SckNoTLSv1

Verschlüsselung ohne Unterstützung von TLS 1.x

Wert 8.388.608 /

0x00800000

SckConnect(),

Siehe SckStartTls(),

SckTlsLow,

SckTlsMed

Option bei den Anweisungen SckConnect() und SckStartTls(), mit der eine verschlüsselte Verbindung ohne Verwendung der TLS 1.x-Protokolle aufgebaut wird. Dies kann beispielsweise bei alten Servern notwendig sein, die mit den TLS-Protokollen Probleme haben. Mit dieser Option wird nur das SSL 3.0-Protokoll unterstützt. Sie kann nur mit SckTlsLow und SckTlsMed kombiniert werden.

Kontakt

_SckOptBind

Verbindung an bestimmte IP-Adresse und / oder bestimmten Port binden

Wert 2.048 /

0x0800

Siehe SckConnect()

Option bei SckConnect(), mit der eine Verbindung an eine bestimmte IP-Adresse und / oder einen bestimmten Port gebunden werden kann.

Dies ist beispielsweise notwendig, wenn ein Computer mehrere IP-Adressen hat und bei der Verbindung zu einem anderen Rechner eine bestimmte IP-Adresse, z. B. durch die Firewall, vorausgesetzt wird.

Kontakt

_SckOptDelay

Schaltet den Nagle-Algorithmus ein

Wert 8 / 0x0008

Siehe SckConnect()

Option bei SckConnect() - Schaltet den Nagle-Algorithmus des TCP/IP-Protokollstacks ein. Damit werden beim Versenden nach Möglichkeit mehrere kleine Datenpakete zu einem größeren Paket zusammengefasst.

Nach dem Verbindungsauflauf werden die TCP-Datensegmente beim Sender so lange zwischengepuffert, bis ein volles Datensegment übertragen werden kann.

Kontakt

_SckOptDontLinger

Schaltet den Linger-Modus aus

Wert 4 / 0x0004

Siehe SckConnect()

Option bei SckConnect() - Schaltet den Linger-Modus des TCP/IP-Protokollstacks aus. Dadurch wird bei einem SckClose() nicht mehr auf das Versenden noch eventuell ungesendeter Daten gewartet.

Kontakt

_SckOptVerify

Überprüfung des serverseitigen Zertifikats

Wert 16.777.216 /
0x01000000

Siehe SckConnect() SckStartTls(),

Option bei den Anweisungen SckConnect() und SckStartTls() - Bei Verwendung einer verschlüsselten Verbindung mit SSL/TLS wird das serverseitige Zertifikat auf Gültigkeit überprüft.



Voraussetzung für eine erfolgreiche Prüfung von Zertifikaten ist eine Sammlung von Stammzertifikaten vertrauenswürdiger Organisationen (root CAs). Diese Sammlung wird als Datei "common\ca-bundle.crt" im CONZEPT 16-Datenverzeichnis abgelegt.

Die Konstante muss mit einer der Konstanten SckTlsLow, SckTlsMed, SckTlsHigh oder SckTlsMax kombiniert werden. Liefert der Server kein gültiges Zertifikat, gibt die Anweisung ErrSckTlsCertificateVerify zurück. Der Fehler bei der Überprüfung kann über die Eigenschaft CertificateError ermittelt werden. Der Fehlerwert kann mittels ErrMapText() in eine Fehlermeldung umgewandelt werden.

Beispiel:

```
tSck # SckConnect(tHost, tPort, _SckTlsHigh | _SckOptVerify);if (tSck = _ErrSckTlsCertificateVeri
```

Kontakt

_SckPortLocal

Lokaler IP-Port

Wert 3

Siehe SckInfo()

Option bei SckInfo() - Den lokalen IP-Port ermitteln.

Kontakt

_SckPortPeer

IP-Port des Zielhosts

Wert 4

Siehe SckInfo()

Option bei SckInfo() - Den IP-Port des Zielhosts ermitteln.

Kontakt

_SckProxySOCKSv4

Version 4 des SOCKS-Proxyserver verwenden Wert 1

/ 0x0001

Siehe SckConnect()

Option bei SckConnect() - Mit der Option _SckProxySOCKSv4 wird angegeben, dass ein SOCKS-Proxyserver der Version 4 verwendet wird.

Bei Verwendung dieser Option, muss im Befehl SckConnect() die IP-Adresse des Proxyservers angegeben werden.

Nach dem Verbindungsaufbau werden die TCP-Datensegmente beim Sender so lange zwischengepuffert, bis ein volles Datensegment übertragen werden kann.

Kontakt

_SckProxySOCKSv4a

Version 4a des SOCKS-Proxyservers verwenden Wert 2

/ 0x0002

Siehe SckConnect()

Option bei SckConnect() - Mit der Option _SckProxySOCKSv4a wird angegeben, dass ein SOCKS-Proxyserver der Version 4a verwendet wird.

Bei Verwendung dieser Option, muss im Befehl SckConnect() die IP-Adresse oder der Name des Proxyservers angegeben werden.

Kontakt

_SckProxySOCKSv5

Version 5 des SOCKS-Proxyservers verwenden Wert 3

/ 0x0003

Siehe SckConnect()

Option bei SckConnect() - Mit der Option _SckProxySOCKSv5 wird angegeben, dass ein SOCKS-Proxyserver der Version 5 verwendet wird.

Bei Verwendung dieser Option, muss im Befehl SckConnect() die IP-Adresse oder der Name des Proxyservers angegeben werden.

Kontakt

_SckReadMax

Lesen bis maximale Länge / Lesen aller empfangener Zeichen Wert 1 /

0x01

Siehe SckRead()

Option bei SckRead() - Es werden soviele Zeichen eingelesen, wie momentan empfangen wurden oder bis die maximale Länge erreicht ist. Die maximale Länge wird entweder durch die übergebene Variable oder durch einen Parameter bei SckRead() bestimmt.

Kontakt

_SckReadyRead

Ermitteln ob Daten zum Lesen vorhanden

Wert 8

Siehe [SckInfo\(\)](#)

Option bei [SckInfo\(\)](#) - Ermitteln ob Daten zum Lesen vorhanden sind.

Das Resultat ist 0, wenn Daten vorhanden sind.

Kontakt

_SckTimeout

Ermitteln oder Setzen des Timeout

Wert 10

Siehe SckInfo()

Option bei SckInfo() - Mit dieser Option kann der Schreib-/Lese-Timeout des Sockets abgefragt (zwei Parameter) oder gesetzt (drei Parameter) werden.



Wird ein neues Timeout zugewiesen und das Resultat von SckInfo() gleichzeitig ausgewertet, wird das neue Timeout zurückgegeben.

Der Timeout wird in Millisekunden angegeben.

Kontakt

_SckTlsHigh

Verschlüsselung mit hohen Sicherheitsanforderungen Wert

100.663.296
/0x06000000

SckConnect(),

SckStartTls(),

Siehe SckTlsLow,

SckTlsMed,

SckTlsMax

Option bei den Anweisungen SckConnect() und SckStartTls(), mit der eine verschlüsselte Verbindung mit hohen Sicherheitsanforderungen aufgebaut wird.

Die hohe Sicherheitsstufe verwendet die TLS-Versionen ab 1.0 und den Schlüsselaustausch mit Diffie-Hellmann (DH) und Elliptic-Curves (ECDH). Die Verschlüsselung erfolgt mit 256- oder 128-Bit.

Die folgenden Cipher-Einstellungen sind definiert:

Protokoll TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0

Schlüsselaustausch ECDHE, DHE

Authentifikation RSA

Verschlüsselung CHACHA20, POLY1305, AES-256, ARIA-256, CAMELLIA-256,
 AES-128, ARIA-128, CAMELLIA-128

Überprüfung AEAD, SHA-2, SHA-1

Die verwendete Protokollversion kann bei bestehender Verbindung mit tSck->SckInfo(_SckVersionTLS) ermittelt werden.

Die unterstützten Cipher-Suiten werden in folgender Reihenfolge angeboten:

Cipher-Suite	Min. Protokoll	Schlüsselaustausch	Authentifikation	Ve
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any	AE (2)
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any	C PO (2)
TLS_AES_128_GCM_SHA256	TLSv1.3	any	any	AE (1)
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AE (2)
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AE (2)
ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	ECDH	RSA	C PO (2)
DHE-RSA-CHACHA20-POLY1305	TLSv1.2	DH	RSA	C PO (2)

Kontakt

DHE-RSA-AES256-CCM8	TLSv1.2	DH	RSA	AE (2)
DHE-RSA-AES256-CCM	TLSv1.2	DH	RSA	AE (2)
ECDHE-ARIA256-GCM-SHA384	TLSv1.2	ECDH	RSA	AR (2)
DHE-RSA-ARIA256-GCM-SHA384	TLSv1.2	DH	RSA	AR (2)
ECDHE-RSA-AES128-GCM-SHA256	TLSv1.2	ECDH	RSA	AE (1)
DHE-RSA-AES128-GCM-SHA256	TLSv1.2	DH	RSA	AE (1)
DHE-RSA-AES128-CCM8	TLSv1.2	DH	RSA	AE (1)
DHE-RSA-AES128-CCM	TLSv1.2	DH	RSA	AE (1)
ECDHE-ARIA128-GCM-SHA256	TLSv1.2	ECDH	RSA	AR (1)
DHE-RSA-ARIA128-GCM-SHA256	TLSv1.2	DH	RSA	AR (1)
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	AE
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	AE
ECDHE-RSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	RSA	Ca
DHE-RSA-CAMELLIA256-SHA256	TLSv1.2	DH	RSA	Ca
ECDHE-RSA-AES128-SHA256	TLSv1.2	ECDH	RSA	AE
DHE-RSA-AES128-SHA256	TLSv1.2	DH	RSA	AE
ECDHE-RSA-CAMELLIA128-SHA256	TLSv1.2	ECDH	RSA	Ca
DHE-RSA-CAMELLIA128-SHA256	TLSv1.2	DH	RSA	Ca
ECDHE-RSA-AES256-SHA	TLSv1	ECDH	RSA	AE
DHE-RSA-AES256-SHA	SSLv3	DH	RSA	AE
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA	Ca
ECDHE-RSA-AES128-SHA	TLSv1	ECDH	RSA	AE
DHE-RSA-AES128-SHA	SSLv3	DH	RSA	AE
DHE-RSA-CAMELLIA128-SHA	SSLv3	DH	RSA	Ca

Kontakt

_SckTlsLow

Verschlüsselung mit niedrigen Sicherheitsanforderungen

Wert 33.554.432 /
0x02000000

SckConnect(),

SckStartTls(),

Siehe SckTlsMed,

SckTlsHigh,

SckTlsMax

Option bei den Anweisungen SckConnect() und SckStartTls(), mit der eine verschlüsselte Verbindung mit niedrigen Sicherheitsanforderungen aufgebaut wird.



Diese Sicherheitsebene sollte nur in den Ausnahmefällen verwendet werden, in denen sonst keine Verbindung mit veralteten Servern hergestellt werden kann.

Die Schlüssellänge für RSA-Schlüssel beträgt mindestens 1024 Bit.

Die folgenden Cipher-Einstellungen sind definiert:

Protokoll	TLS 1.2, TLS 1.1, TLS 1.0, SSL 3.0
Schlüsselaustausch	ECDHE, DHE, RSA
Authentifikation	ECDSA, RSA, DSS
Verschlüsselung	CHACHA20, POLY1305, AES-256, ARIA-256, CAMELLIA-256, AES-128, ARIA-128, CAMELLIA-128, SEED, IDEA
Überprüfung	AEAD, SHA-2, SHA-1

Die verwendete Protokollversion kann bei bestehender Verbindung mit tSck->SckInfo(SckVersionTLS) ermittelt werden.

Die unterstützten Cipher-Suiten werden in folgender Reihenfolge angeboten:

Cipher-Suite	Min. Protokoll	Schlüsselaustausch	Authentifikation
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any
TLS_AES_128_GCM_SHA256	TLSv1.3	any	any
ECDHE-ECDSA-AES256-GCM-SHA384	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA
DHE-DSS-AES256-GCM-SHA384	TLSv1.2	DH	DSS
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA

Kontakt			
ECDHE-ECDSA-CHACHA20-POLY1305	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	ECDH	RSA
DHE-RSA-CHACHA20-POLY1305	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES256-CCM8	TLSv1.2	ECDH	ECDSA
ECDHE-ECDSA-AES256-CCM	TLSv1.2	ECDH	ECDSA
DHE-RSA-AES256-CCM8	TLSv1.2	DH	RSA
DHE-RSA-AES256-CCM	TLSv1.2	DH	RSA
ECDHE-ECDSA-ARIA256-GCM-SHA384 TLSv1.2		ECDH	ECDSA
ECDHE-ARIA256-GCM-SHA384	TLSv1.2	ECDH	RSA
DHE-DSS-ARIA256-GCM-SHA384	TLSv1.2	DH	DSS
DHE-RSA-ARIA256-GCM-SHA384	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES128-GCM-SHA256	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES128-GCM-SHA256	TLSv1.2	ECDH	RSA
DHE-DSS-AES128-GCM-SHA256	TLSv1.2	DH	DSS
DHE-RSA-AES128-GCM-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES128-CCM8	TLSv1.2	ECDH	ECDSA
ECDHE-ECDSA-AES128-CCM	TLSv1.2	ECDH	ECDSA
DHE-RSA-AES128-CCM8	TLSv1.2	DH	RSA
DHE-RSA-AES128-CCM	TLSv1.2	DH	RSA
ECDHE-ECDSA-ARIA128-GCM-SHA256 TLSv1.2		ECDH	ECDSA
ECDHE-ARIA128-GCM-SHA256	TLSv1.2	ECDH	RSA
DHE-DSS-ARIA128-GCM-SHA256	TLSv1.2	DH	DSS

Kontakt

DHE-RSA-ARIA128-GCM-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES256-SHA384	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA
DHE-DSS-AES256-SHA256	TLSv1.2	DH	DSS
ECDHE-ECDSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	RSA
DHE-RSA-CAMELLIA256-SHA256	TLSv1.2	DH	RSA
DHE-DSS-CAMELLIA256-SHA256	TLSv1.2	DH	DSS
ECDHE-ECDSA-AES128-SHA256	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES128-SHA256	TLSv1.2	ECDH	RSA
DHE-RSA-AES128-SHA256	TLSv1.2	DH	RSA
DHE-DSS-AES128-SHA256	TLSv1.2	DH	DSS
ECDHE-ECDSA-CAMELLIA128-SHA256	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-CAMELLIA128-SHA256	TLSv1.2	ECDH	RSA
DHE-RSA-CAMELLIA128-SHA256	TLSv1.2	DH	RSA
DHE-DSS-CAMELLIA128-SHA256	TLSv1.2	DH	DSS
ECDHE-ECDSA-AES256-SHA	TLSv1	ECDH	ECDSA
ECDHE-RSA-AES256-SHA	TLSv1	ECDH	RSA
DHE-RSA-AES256-SHA	SSLv3	DH	RSA
DHE-DSS-AES256-SHA	SSLv3	DH	DSS
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA
DHE-DSS-CAMELLIA256-SHA	SSLv3	DH	DSS
ECDHE-ECDSA-AES128-SHA	TLSv1	ECDH	ECDSA
ECDHE-RSA-AES128-SHA	TLSv1	ECDH	RSA
DHE-RSA-AES128-SHA	SSLv3	DH	RSA
DHE-DSS-AES128-SHA	SSLv3	DH	DSS
DHE-RSA-CAMELLIA128-SHA	SSLv3	DH	RSA
DHE-DSS-CAMELLIA128-SHA	SSLv3	DH	DSS
AES256-GCM-SHA384	TLSv1.2	RSA	RSA
AES256-CCM8	TLSv1.2	RSA	RSA
AES256-CCM	TLSv1.2	RSA	RSA
ARIA256-GCM-SHA384	TLSv1.2	RSA	RSA
AES128-GCM-SHA256	TLSv1.2	RSA	RSA
AES128-CCM8	TLSv1.2	RSA	RSA

Kontakt

AES128-CCM	TLSv1.2	RSA	RSA
ARIA128-GCM-SHA256	TLSv1.2	RSA	RSA
AES256-SHA256	TLSv1.2	RSA	RSA
CAMELLIA256-SHA256	TLSv1.2	RSA	RSA
AES128-SHA256	TLSv1.2	RSA	RSA
CAMELLIA128-SHA256	TLSv1.2	RSA	RSA
AES256-SHA	SSLv3	RSA	RSA
CAMELLIA256-SHA	SSLv3	RSA	RSA
AES128-SHA	SSLv3	RSA	RSA
CAMELLIA128-SHA	SSLv3	RSA	RSA
DHE-RSA-SEED-SHA	SSLv3	DH	RSA
DHE-DSS-SEED-SHA	SSLv3	DH	DSS
SEED-SHA	SSLv3	RSA	RSA
IDEA-CBC-SHA	SSLv3	RSA	RSA

Kontakt

_SckTlsMax

**Verschlüsselung mit maximalen Sicherheitsanforderungen Wert
134.217.728 /**

0x08000000

SckConnect(),

SckStartTls(),

Siehe SckTlsLow,

SckTlsMed,

SckTlsHigh

Option bei den Anweisungen SckConnect() und SckStartTls(), mit der eine verschlüsselte Verbindung mit maximalen Sicherheitsanforderungen aufgebaut wird.

Die maximale Sicherheitsstufe unterstützen aktuell nur wenige SSL-Clients und -Server. Es wird ausschließlich die TLS-Version 1.2 verwendet, Forward Secrecy ist obligatorisch. Daher geht der Schlüsselaustausch nur mit Diffie-Hellman (DH), optional mit Elliptic Curves (ECDH). Es wird ausschließlich mit 256 Bit verschlüsselt.

Die folgenden Cipher-Einstellungen sind definiert:

Protokoll TLS 1.3, TLS 1.2

Schlüsselaustausch ECDHE, DHE

Authentifikation RSA

Verschlüsselung CHACHA20, POLY1305, AES-256, ARIA-256, CAMELLIA-256

Überprüfung AEAD, SHA-2, SHA-1

Beim Schlüsselaustausch mit Diffie-Hellman werden im Clientmodus die Längen 512, 1024, 2048 und 4096 Bit unterstützt. Im Servermodus (SOA-Service) wird eine Schlüssellänge von 2048 Bit verwendet.

Die verwendete Protokollversion kann bei bestehender Verbindung mit tSck->SckInfo(SckVersionTLS) ermittelt werden.

Die unterstützten Cipher-Suiten werden in folgender Reihenfolge angeboten:

Cipher-Suite	Min. Protokoll	Schlüsselaustausch	Authentifikation	Ve
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any	AE (2)
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any	C PO (2)
TLS_AES_128_GCM_SHA256	TLSv1.3	any	any	AE (1)
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AE (2)
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AE (2)
ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	ECDH	RSA	C PO (2)

Kontakt

DHE-RSA-CHACHA20-POLY1305	TLSv1.2	DH	RSA	(2 C PO
DHE-RSA-AES256-CCM8	TLSv1.2	DH	RSA	(2 AE
DHE-RSA-AES256-CCM	TLSv1.2	DH	RSA	(2 AE
ECDHE-ARIA256-GCM-SHA384	TLSv1.2	ECDH	RSA	(2 AR
DHE-RSA-ARIA256-GCM-SHA384	TLSv1.2	DH	RSA	(2 AR
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	(2 AE
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	AE
ECDHE-RSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	RSA	Ca
DHE-RSA-CAMELLIA256-SHA256	TLSv1.2	DH	RSA	Ca
ECDHE-RSA-AES256-SHA	TLSv1.2	ECDH	RSA	AE
DHE-RSA-AES256-SHA	SSLv3	DH	RSA	AE
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA	Ca

Kontakt

_SckTlsMed

Verschlüsselung mit durchschnittlichen Sicherheitsanforderungen

Wert 67.108.864 /
0x04000000

SckConnect(),

SckStartTls(),

Siehe SckTlsLow,

SckTlsHigh,

SckTlsMax

Option bei den Anweisungen SckConnect() und SckStartTls(), mit der eine verschlüsselte Verbindung mit mittleren Sicherheitsanforderungen aufgebaut wird.

Zusätzlich zu den bei SckTlsHigh unterstützten Verfahren wird für die Kompatibilität mit älteren Versionen (beispielsweise Windows XP mit IE6) auch das inzwischen veraltete Protokoll SSL 3.0 und die Verschlüsselung mit RSA, welche kein Forward Secrecy unterstützt, erlaubt.

Die Schlüssellänge für RSA-Schlüssel beträgt mindestens 1024 Bit.

Die folgenden Cipher-Einstellungen sind definiert:

Protokoll TLS 1.3, TLS 1.2, TLS 1.1, TLS 1.0, SSL 3.0

Schlüsselaustausch ECDHE, DHE, RSA

Authentifikation ECDSA, RSA

Verschlüsselung CHACHA20, POLY1305, AES-256, ARIA-256, CAMELLIA-256,
AES-128, ARIA-128, CAMELLIA-128, SEED

Überprüfung AEAD, SHA-2, SHA-1

Die verwendete Protokollversion kann bei bestehender Verbindung mit tSck->SckInfo(SckVersionTLS) ermittelt werden.

Die unterstützten Cipher-Suiten werden in folgender Reihenfolge angeboten:

Cipher-Suite	Min. Protokoll	Schlüsselaustausch	Authentifikation
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any
TLS_AES_128_GCM_SHA256	TLSv1.3	any	any
ECDHE-ECDSA-AES256-GCM-SHA384	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA
ECDHE-ECDSA-CHACHA20-POLY1305	TLSv1.2	ECDH	ECDSA

Kontakt

ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	ECDH	RSA
DHE-RSA-CHACHA20-POLY1305	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES256-CCM8	TLSv1.2	ECDH	ECDSA
ECDHE-ECDSA-AES256-CCM	TLSv1.2	ECDH	ECDSA
DHE-RSA-AES256-CCM8	TLSv1.2	DH	RSA
DHE-RSA-AES256-CCM	TLSv1.2	DH	RSA
ECDHE-ECDSA-ARIA256-GCM-SHA384 TLSv1.2		ECDH	ECDSA
ECDHE-ARIA256-GCM-SHA384	TLSv1.2	ECDH	RSA
DHE-RSA-ARIA256-GCM-SHA384	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2		ECDH	ECDSA
ECDHE-RSA-AES128-GCM-SHA256	TLSv1.2	ECDH	RSA
DHE-RSA-AES128-GCM-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES128-CCM8	TLSv1.2	ECDH	ECDSA
ECDHE-ECDSA-AES128-CCM	TLSv1.2	ECDH	ECDSA
DHE-RSA-AES128-CCM8	TLSv1.2	DH	RSA
DHE-RSA-AES128-CCM	TLSv1.2	DH	RSA
ECDHE-ECDSA-ARIA128-GCM-SHA256 TLSv1.2		ECDH	ECDSA
ECDHE-ARIA128-GCM-SHA256	TLSv1.2	ECDH	RSA
DHE-RSA-ARIA128-GCM-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES256-SHA384	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA

Kontakt

ECDHE-ECDSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	RSA
DHE-RSA-CAMELLIA256-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES128-SHA256	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-AES128-SHA256	TLSv1.2	ECDH	RSA
DHE-RSA-AES128-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-CAMELLIA128-SHA256	TLSv1.2	ECDH	ECDSA
ECDHE-RSA-CAMELLIA128-SHA256	TLSv1.2	ECDH	RSA
DHE-RSA-CAMELLIA128-SHA256	TLSv1.2	DH	RSA
ECDHE-ECDSA-AES256-SHA	TLSv1	ECDH	ECDSA
ECDHE-RSA-AES256-SHA	TLSv1	ECDH	RSA
DHE-RSA-AES256-SHA	SSLv3	DH	RSA
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA
ECDHE-ECDSA-AES128-SHA	TLSv1	ECDH	ECDSA
ECDHE-RSA-AES128-SHA	TLSv1	ECDH	RSA
DHE-RSA-AES128-SHA	SSLv3	DH	RSA
DHE-RSA-CAMELLIA128-SHA	SSLv3	DH	RSA
AES256-GCM-SHA384	TLSv1.2	RSA	RSA
 AES256-CCM8	 TLSv1.2	 RSA	 RSA
 AES256-CCM	 TLSv1.2	 RSA	 RSA
 ARIA256-GCM-SHA384	 TLSv1.2	 RSA	 RSA
 AES128-GCM-SHA256	 TLSv1.2	 RSA	 RSA
 AES128-CCM8	 TLSv1.2	 RSA	 RSA
 AES128-CCM	 TLSv1.2	 RSA	 RSA
 ARIA128-GCM-SHA256	 TLSv1.2	 RSA	 RSA
 AES256-SHA256	 TLSv1.2	 RSA	 RSA
 CAMELLIA256-SHA256	 TLSv1.2	 RSA	 RSA
 AES128-SHA256	 TLSv1.2	 RSA	 RSA
 CAMELLIA128-SHA256	 TLSv1.2	 RSA	 RSA
 AES256-SHA	 SSLv3	 RSA	 RSA
 CAMELLIA256-SHA	 SSLv3	 RSA	 RSA
 AES128-SHA	 SSLv3	 RSA	 RSA
 CAMELLIA128-SHA	 SSLv3	 RSA	 RSA
 DHE-RSA-SEED-SHA	 SSLv3	 DH	 RSA

Kontakt

SckTlsSNI

Server Name Indication beim Aufbau der verschlüsselten Verbindung verwenden Wert **268.435.456**

/ **0x10000000**

SckConnect(),

Siehe SckStartTls(),

MailOpen()

Option bei der Anweisung SckConnect(), SckStartTls() und MailOpen() mit der beim verschlüsselten Verbindungsauftbau Server Name Indication (SNI) verwendet wird.

Die Konstante muss mit einer der Konstanten SckTlsLow, SckTlsMed, SckTlsHigh oder SckTlsMax kombiniert werden, damit sie sich auswirkt.

Kontakt

_SckVersionTLS

TLS-Version bei verschlüsselter Verbindung

Wert 12

Siehe SckInfo()

Option bei SckInfo() - Es wird die verwendete TLS-Version gefolgt von einem Leerzeichen und der Cipher-Suite bei einer verschlüsselten Verbindung zurückgegeben. Folgende Versionen können zurückgegeben werden.

" keine verschlüsselte Verbindung

'SSLv3' Verschlüsselt mit SSL 3.0

'TLSv1' Verschlüsselt mit TLS 1.0

'TLSv1.1' Verschlüsselt mit TLS 1.1

'TLSv1.2' Verschlüsselt mit TLS 1.2

Beispiele:

TLsv1 DHE-RSA-AES256-SHA TLSv1.2 DHE-RSA-AES256-GCM-SHA384

Kontakt

_SckVol

Übertragene Datenvolumen in Bytes

Wert 5

Siehe SckInfo()

Option bei SckInfo() - Das übertragene Datenvolumen in Bytes ermitteln.

Kontakt

_SckVolRead

Empfangene Datenvolumen in Bytes

Wert 6

Siehe SckInfo()

Option bei SckInfo() - Das empfangene Datenvolumen in Bytes ermitteln.

Kontakt

_SckVolWrite

Gesendete Datenvolumen in Bytes

Wert 7

Siehe SckInfo()

Option bei SckInfo() - Das gesendete Datenvolumen in Bytes ermitteln.

Kontakt

Message-Exchange-Befehle

Befehle für den Datenaustausch zwischen CONZEPT 16-Clients

Verwandte

Befehle, Liste

sortiert nach

Gruppen,

Siehe Alphabetische

Liste aller

Befehle,

Aufbau (Blog),

Befehle (Blog)

Befehle

- MsxClose
- MsxOpen
- MsxRead
- MsxReadMem
- MsxWrite
- MsxWriteMem

Konstanten

- MsxData
- MsxEnd
- MsxFile
- MsxItem
- MsxMessage
- MsxRead
- MsxSocket
- MsxWrite

Kommunikation über Message-Exchange

Die MSX-Befehle ermöglichen den Datenaustausch über ein selbst zu definierendes Protokoll. Dabei ist das Medium der Datenübertragung nicht relevant. CONZEPT 16 unterstützt den Datenaustausch über TCP/IP-Sockets und externe Dateien. Die Kommunikation über Sockets erfolgt dabei synchron, da nur dann eine Verbindung aufgebaut werden kann, wenn ein Empfänger Daten auf einen bestimmten Port erwartet (siehe SckListen()). Die Kommunikation über eine externe Datei erfolgt asynchron. Ein gleichzeitiges Lesen und Schreiben ist nicht möglich. Dafür kann eine Datei geschrieben und erst zu einem späteren Zeitraum gelesen werden.

Mit den MSX-Befehlen kann der Entwickler Datenkanäle öffnen und anschließend Nachrichten in beliebiger Anzahl und Größe darüber senden oder lesen. Nachrichten setzen sich aus einer unbeschränkten Anzahl an Elementen zusammen, die wiederum mehrere Datenfelder enthalten. Diese Datenfelder sind nicht an einen speziellen Datentyp gebunden.

Die einzelnen Nachrichten werden mit IDs versehen, wodurch es möglich ist, unbekannte Nachrichtentypen zu ignorieren. Die Elemente versieht der Entwickler ebenfalls mit IDs. Durch diese Identifizierung können beispielsweise ebenfalls

Kontakt

unbekannte oder bereits übertragene Daten übersprungen werden. Die in den Elementen enthaltenen Datenfelder werden seriell geschrieben und ausgelesen. Neue Datenfelder können jederzeit hinzugefügt werden, ohne das der Empfänger diese bereits kennen muss. In diesem Fall werden die neuen Felder durch die Programmierung ignoriert und das nächste Element wird gelesen. Bei einer Erweiterung des Protokolls müssen daher nicht zwingendermaßen alle beteiligten Komponenten aktualisiert werden. Die MSX-Befehle ermöglichen somit nicht nur das Erstellen von abwärts-, sondern auch von aufwärtskompatiblen Protokollen.



In der Datenbank "CodeLibrary.ca1" befindet sich ein Beispiel zur Verwendung der MSX-Befehle.

Kontakt

obj -> MsxClose()  

Nachrichtenkanal schließen

obj Deskriptor des

Nachrichtenkanals

Siehe [Verwandte Befehle](#),
[MsxOpen\(\)](#)

Mit dieser Anweisung wird ein mit [MsxOpen\(\)](#) angelegter Nachrichtenkanal wieder entfernt. Als (obj) wird der Deskriptor des Nachrichtenkanals übergeben. Anschließend ist der Deskriptor nicht mehr gültig.

Der zugrundeliegende Datei- oder Socketdeskriptor bleibt offen und muss mit [FsiClose\(\)](#) oder [SckClose\(\)](#) geschlossen werden.

Beispiel:

```
tMsx # MsxOpen(_MsxSocket | _MsxRead, tSck);...if (tMsx > 0){ ... tMsx->MsxClose();}...
```

Mögliche Laufzeitfehler:

[ErrHdlInvalid](#) Der übergebene Deskriptor in (obj) ist ungültig.

Kontakt

MsxOpen(int1, handle2)



: handle Nachrichtenkanal
öffnen

Modus

- [MsxSocket](#) Kommunikation über Socket-Verbindung
- [MsxFile](#) Kommunikation über externe Datei
- [MsxThread](#) Kommunikation mit einem als Thread gestarteten Job
- [MsxProcess](#) Kommunikation mit einem als Prozess gestarteten Job
- [MsxRead](#) Lesender Zugriff
- [MsxWrite](#) Schreibender Zugriff

int1

handle2 Deskriptor des Sockets, der Datei, des Tasks oder des Jobs

Resultat [handle](#) Deskriptor des Nachrichtenkanals

Siehe [Verwandte Befehle](#),
[MsxClose\(\)](#)

Mit diesem Befehl wird ein Datenaustauschkanal geöffnet, welcher das Schreiben oder Lesen von Nachrichten in beliebiger Anzahl und Größe ermöglicht. Die Kommunikation kann entweder über eine Socket-Verbindung, über eine externe Datei oder über den Prozessspeicher hergestellt werden. Die Art der Verbindung ([MsxSocket](#) / [MsxFile](#) / [MsxThread](#) / [MsxProcess](#)) und ob Daten geschrieben ([MsxWrite](#)) oder gelesen ([MsxRead](#)) werden sollen, wird im Parameter (int1) angegeben.

In (handle2) wird der entsprechende Deskriptor (Socket-Verbindung, geöffnete Datei oder Job- bzw. JobControl-Objekt) übergeben. Es kann nur eine Konstante für die Verbindung mit einer Konstanten für die Datenrichtung kombiniert werden.

Die Kommunikation zwischen einem Client bzw. einer Ereignisfunktion des SOA-Service und einem als Thread gestartetem Job oder zwischen zwei als Thread gestarteten Jobs erfolgt über den Prozessspeicher.

Zum Verbindungsauftbau aus der Ereignisfunktion oder einem Job, muss das JobControl-Objekt des zu erreichenden Jobs übergeben werden. Zur Verbindungsannahme im Job, muss das Task-Objekt des Jobs übergeben werden.

Zum Erzeugen des Kanals muss in der Ereignisfunktion der Deskriptor des JobControl-Objekts übergeben werden. In einer mit JobStart() gestarteten Funktion muss der Kanal mit dem übergebenen Task-Objekt geöffnet werden.

Kontakt

Außer bei einem Kanal über eine externe Datei, kann ein Kanal zum Senden (Schreiben) und zum Empfangen (Lesen) von Nachrichten verwendet werden. In diesem Fall müssen dann zwei Kanäle mit dem selben Deskriptor geöffnet werden.

Um Nachrichten über Sockets zu empfangen, muss beim Empfänger ein Nachrichtenkanal zum Lesen geöffnet werden. Dazu kann ein SckListen() durchgeführt werden. Im darauf folgenden EvtSocket können die Daten dann abgeholt werden. Falls die Nachrichtenübertragung über eine Datei durchgeführt wird, kann der Empfänger das Dateiverzeichnis entweder mit FsiMonitorAdd() überwachen oder den Empfang zu einem selbst definierten Zeitpunkt durchführen. Beim Überwachen des Verzeichnisses wird ebenfalls ein Ereignis, das EvtFsiMonitor, ausgelöst.

Beispiele:

```
// Nachricht über Socket versenden tSck # SckConnect('10.1.1.16', 1250); if (tSck > 0) { tMsx # Msx
```

Folgende Laufzeitfehler sind möglich:

ErrHdlInvalid Der übergebene Socket- oder Dateideskriptor ist ungültig.

ErrValueInvalid Es wurde in (int1) eine ungültige Konstantenkombination übergeben.

Kontakt

obj -> MsxRead(int1, var2[,
int3]) : int



Nachrichtenkanal lesen

obj Deskriptor des Nachrichtenkanals

Nachrichtenbereich

MsxMessage Nachrichten-ID

int1 MsxItem Bereichs-ID

MsxData Datenbereich

MsxEnd Ende der Nachricht

var2 Inhalt

int3 Länge (optional)

Fehlerwert

ErrOK Lesen erfolgreich

ErrEndOfFile Keine weiteren

Nachrichten (bei
MsxFile)

Resultat int ErrData Allgemeiner



Datenfehler

Socket-Fehler

ErrSck... Fehler externe

Dateioperationen

Verwandte Befehle, MsxOpen(),

Siehe

MsxWrite()

Diese Anweisung liest die verschiedenen Bereiche einer Nachricht. Eine Nachricht kann aus folgenden Bereichen bestehen:

- MsxMessage

Damit wird eine neue Nachricht geöffnet. In (var2) muss eine Variable vom Typ int übergeben werden, in der anschließend die Nachrichten-ID steht. Anhand der ID kann ein benutzerdefinierter Nachrichtentyp ermittelt werden. Eine bereits geöffnete Nachricht wird durch diese Operation automatisch geschlossen.

Sind keine weiteren Nachrichten vorhanden wird dies über den Rückgabewert der Funktion übermittelt. Dabei werden die Fehlerkonstanten des entsprechenden Mediums verwendet. Bei der Verwendung von Sockets wird ErrTimeout zurückgegeben, wenn keine weiteren Daten vorhanden sind und ErrSckRead, wenn die Verbindung beendet wurde. Falls in einer externen Datei keine weiteren Daten vorhanden sind, ist das Resultat ErrEndOfFile.

- MsxItem

Damit wird ein neues Element geöffnet. In (var2) muss eine Variable vom Typ int angegeben werden, in der anschließend die Item-ID steht. Anhand der ID kann ein benutzerdefinierter Elementtyp ermittelt werden. Ein bereits geöffnetes Element wird durch diese Operation automatisch geschlossen, eventuell noch nicht gelesene Daten werden dabei übersprungen. MsxItem kann nur verwendet werden, wenn zuvor eine Nachricht geöffnet wurde. Falls

Kontakt

kein weiteres Element mehr vorhanden ist, wird eine Item-ID von 0 zurückgeliefert, wobei das Funktionsresultat ErrOk ist.

- MsxData

Damit werden Daten aus einem Element gelesen. In (var2) wird eine Variable des erforderlichen Typs übergeben. Optional kann in (int3) eine maximale Datenlänge angegeben werden, die die Anzahl der in die Variable zu übertragenen Bytes beim Datentyp alpha beschränkt. Alle Daten werden seriell gelesen. Dabei müssen nicht zwingend alle Datenfelder ausgelesen werden. MsxData wird nur bei einem offenen Element verwendet.

- MsxEnd

Damit wird die offene Nachricht geschlossen, eventuell noch nicht gelesene Daten oder Elemente werden übersprungen.

Eine Nachricht kann immer nur eine Nachrichten-ID besitzen. Die Nachrichten-ID 0 ist dabei unzulässig. IDs von Items dürfen sich wiederholen. Auch hier ist die ID 0 nicht zulässig. Der eigentliche Nachrichteninhalt befindet sich in einem oder mehreren Datenfeldern pro Item. Beim Lesen von Nachrichten können einzelne Items übersprungen werden. Es wird dann einfach das nächste Item gelesen.

Üblicherweise erfolgt die Auswertung einer Nachricht in einem switch-Konstrukt.
Wobei die verschiedenen Abschnitte einzeln verarbeitet werden.

Beispiel:

```
tMsx # MsxOpen(_MsxSocket | _MsxRead, aHandle);if (tMsx > 0){ try { tMsx->MsxRead(_MsxMessage,
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebene Deskriptor in (obj) ist ungültig.

ErrFldType Die Variable oder das Feld in (var2) hat nicht den passenden Datentyp.

ErrValueInvalid In (int1) wurde ein falscher Wert übergeben.

Kontakt

obj -> MsxReadMem(handle1, int2, int3) :



int

Nachrichtenkanal in Memory-Objekt lesen

Deskriptor eines

obj

Nachrichtenkanals

handle1 Deskriptor eines

Memory-Objekts

Zielposition im

int2

Memory-Objekt

Anzahl der zu

int3

lesenden Bytes

Resultat int Fehlerwert



Verwandte Befehle,

Siehe

[MsxWriteMem\(\)](#)

Dieser Befehl liest binäre Daten aus dem Nachrichtenkanal (obj) in das **Memory-Objekt** (handle1) ein.

Die Funktion entspricht damit [MsxRead\(MsxData, ...\)](#). Vor dem Aufruf von MsxReadMem() muss ein Nachrichtenelement bereits mit [MsxRead\(MsxItem, ...\)](#) geöffnet sein. In (int2) wird die Zielposition im Memory-Objekt und in (int3) die Datenlänge angegeben. Die Datenlänge muss identisch mit der beim Schreiben angegebenen Länge sein (siehe [MsxWriteMem\(\)](#)).

Das Resultat enthält den Fehlerwert oder [ErrOk](#), wenn kein Fehler aufgetreten ist.

Mögliche Laufzeitfehler:

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

[ErrValueRange](#) Der in (int2) oder (int3) übergebene Wert ist außerhalb des zulässigen Bereichs.

Kontakt

obj -> MsxWrite(int1, var2[,



int3]) : int

Nachrichteninhalt schreiben

 Deskriptor eines

obj Nachrichtenkanals

 Nachrichtenbereich

MsxMessage Nachrichten-ID

MsxItem Bereichs-ID

int1 MsxData Datenbereich

MsxEnd Nachrichtenpuffer

 abschließen

var2 Variable oder Konstante mit

dem zu schreibendem Inhalt

int3 Länge des Inhalts (optional)

Resultat int Fehlerwert



Verwandte Befehle,
 MsxOpen(), MsxRead()

Siehe

Socket-Fehler, Fehler externe
 Dateioperationen

Diese Anweisung schreibt die Nachrichtenteile in den Nachrichtenkanal. Als Objekt wird der Deskriptor des Nachrichtenkanals, der von MsxOpen() zurückgegeben wurde, übergeben. Folgende Nachrichtenteile können geschrieben werden:

- MsxMessage

Damit wird eine neue Nachricht geöffnet. In (var2) muss ein Wert vom Typ **int** ungleich 0 übergeben werden, der als Nachrichten-ID übermittelt wird. Anhand der ID kann ein benutzerdefinierter Nachrichtentyp definiert werden. Eine bereits geöffnete Nachricht wird durch diese Operation automatisch geschlossen.

- MsxItem

Damit wird ein neues Element geöffnet. In (var2) muss ein Wert vom Typ **int** ungleich 0 angegeben werden, der als Item-Id übermittelt wird. Anhand der ID kann ein benutzerdefinierter Elementtyp definiert werden. Ein bereits geöffnetes Element wird durch diese Operation automatisch geschlossen. MsxItem kann nur verwendet werden, wenn zuvor eine Nachricht geöffnet wurde.

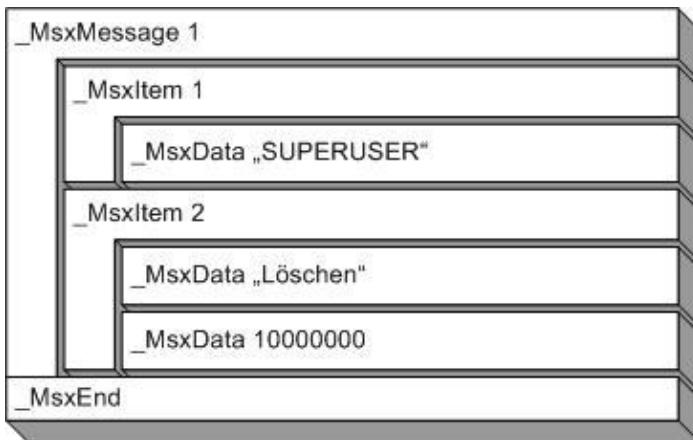
- MsxData

Damit werden Daten in ein Element geschrieben. In (var2) werden die zu schreibenden Daten übergeben. In (int3) kann die maximale Datenlänge angegeben werden, die die Anzahl der zu schreibenden Bytes bei Werten vom Typ **alpha** beschränkt. Alle Datenfelder werden seriell geschrieben. MsxData kann nur bei einem offenen Element verwendet werden.

- MsxEnd

Damit wird die offene Nachricht geschlossen.

Kontakt



Der Rückgabewert ist rOk, wenn die Operation erfolgreich war.

Beispiel:

```
tSck # SckConnect('10.1.1.16', 12500); tMsx # MsxOpen(_MsxSocket | _MsxWrite, tSck); ... try{ tMsx-
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der übergebene Deskriptor in (obj) ist ungültig.

ErrFldType Der Wert in (var2) hat nicht den passenden Datentyp.

ErrValueInvalid In (int1) oder (var2) wurde ein falscher Wert übergeben.

Kontakt

obj -> MsxWriteMem(handle1, int2, int3) : int |  Nachrichtenkanal

schreiben aus einem Memory-Objekt

obj Deskriptor eines Nachrichtenkanals

handle1 Deskriptor eines Memory-Objekts

int2 Startposition im Memory-Objekt

int3 Anzahl der zu schreibenden Bytes

Resultat int Anzahl der geschriebenen Bytes oder Fehlerwert

Siehe [Verwandte Befehle, MsxReadMem\(\)](#)

Dieser Befehl schreibt binäre Daten aus dem Memory-Objekt (handle1) in den Nachrichtenkanal (obj). Die Funktion entspricht damit MsxWrite(MsxData, ...). Vor dem Aufruf von

MsxWriteMem() muss ein Nachrichtenelement bereits mit MsxWrite(MsxItem, ...) geöffnet sein.

In (int2) wird die Startposition im Memory-Objekt und in (int3) die Datenlänge angegeben. Die Funktion erzeugt im Nachrichten-Stream ein binäres Feld mit der Länge (int3). Zum Einlesen der Daten dieses binären Felds muss exakt die gleiche Länge bei MsxReadMem() angegeben werden.

Das Resultat enthält den Fehlerwert oder ErrOk, wenn kein Fehler aufgetreten ist.

Mögliche Laufzeitfehler:

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

ErrValueRange Der in (int2) oder (int3) angegebene wert ist außerhalb des zulässigen Bereichs.

Kontakt

Konstanten für Message Exchange-Befehle

Siehe [Alle Befehle](#)

<u>MsxData</u>	Datenbereich lesen/schreiben
<u>MsxEnd</u>	Nachricht abschließen
<u>MsxFile</u>	Kommunikation über externe Datei
<u>MsxItem</u>	Item-ID lesen/schreiben
<u>MsxMessage</u>	Nachrichten-ID lesen/schreiben
<u>MsxRead</u>	Nachricht zum Lesen öffnen
<u>MsxSocket</u>	Kommunikation über Socket
<u>MsxThread</u>	Kommunikation mit einem Thread
<u>MsxWrite</u>	Nachricht zum Schreiben öffnen

Kontakt

_MsxData

Datenbereich lesen/schreiben

Wert 3 / 0x03

Verwandte

Siehe Befehle,

MsxRead(),

MsxWrite()

Option der Befehle MsxRead() und MsxWrite() - Es wird ein Datenfeld gelesen, beziehungsweise geschrieben.

Kontakt

_MsxEnd

Nachricht abschließen

Wert 4 / 0x04

Verwandte

Siehe Befehle,

MsxWrite(),

MsxRead()

Option der Befehle MsxWrite() und MsxRead() - Das Nachrichtenende wird gelesen, beziehungsweise geschrieben.

Kontakt

_MsxFile

Kommunikation über externe Datei

Wert 1 / 0x01

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Die Kommunikation findet über eine externe Datei statt.

Kontakt

_MsxItem

Item-ID lesen/schreiben

Wert 2 / 0x02

Verwandte

Siehe Befehle,

MsxRead(),

MsxWrite()

Option der Befehle MsxRead() und MsxWrite() - Es wird ein neues Element geöffnet und die Element-ID wird gelesen, beziehungsweise geschrieben.

Kontakt

_MsxMessage

Message-ID lesen/schreiben

Wert 1 / 0x01

Verwandte

Siehe Befehle,

MsxRead(),

MsxWrite()

Option der Befehle MsxRead() und MsxWrite() - Es wird eine neue Nachricht geöffnet und die Nachrichten-ID gelesen, beziehungsweise geschrieben.

Kontakt

_MsxRead

Nachricht zum Lesen öffnen

Wert 16 / 0x10

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Der Nachrichtenpuffer wird zum Lesen geöffnet.

Anschließend kann eine Nachricht mit MsxRead()-Anweisungen ausgelesen werden.

Kontakt

_MsxSocket

Kommunikation über Socket

Wert 2 / 0x02

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Die Kommunikation findet über eine Socket-Verbindung statt.

Kontakt

_MsxThread

Kommunikationskanal mit Thread einrichten Wert

4 / 0x04

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Es wird ein Nachrichtenkanal mit einem anderen Thread eingerichtet. Die Kommunikation kann sowohl zwischen dem Client bzw. der Ereignisfunktion des SOA-Service und einem als Thread gestarteten Job, als auch zwischen zwei als Thread gestarteten Jobs hergestellt werden.

Kontakt

_MsxWrite

Nachricht schreiben

Wert 32 / 0x20

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Der Nachrichtenpuffer wird zum Schreiben geöffnet.

Anschließend kann eine Nachricht mit MsxWrite()-Anweisungen geschrieben werden.

TAPI-Befehle

Befehle für die TAPI-Schnittstelle

Verwandte

Befehle, Liste

sortiert nach

Siehe Gruppen,

Alphabetische

Liste aller

Befehle

Befehle

- [TapiCall](#)
- [TapiClose](#)
- [TapiConference](#)
- [TapiConferenceCommit](#)
- [TapiConferenceDial](#)
- [TapiDial](#)
- [TapiForward](#)
- [TapiInfo](#)
- [TapiListen](#)
- [TapiOpen](#)
- [TapiPickup](#)

Mit den TAPI-Befehlen können die Funktionen der TAPI-Schnittstelle (Telephony API) benutzt werden. Die TAPI-Schnittstelle teilt sich in zwei Ebenen: Die Schnittstelle des Betriebssystems und die Gerätetreiber ([TapiDevice](#)).

Mit dem Befehl [TapiOpen\(\)](#) wird die Schnittstelle initialisiert und der Deskriptor einer Liste mit Gerätetreibern zurückgegeben. Die Liste der Gerätetreiber und die Gerätetreiber selber können verwendet werden, bis mit dem Befehl [TapiClose\(\)](#) die Liste geschlossen wird.

Um eine Nummer zu wählen, wird der Befehl [TapiDial\(\)](#) verwendet. Das Format des zu übergebenen Wählstrings hängt im wesentlichen von der Implementation des Gerätetreibers ab. In der Regel sollten hier nur Zeichenketten übergeben werden, die ausschließlich aus Ziffern bestehen.

Sollen eingehende Telefonate erkannt werden, wird der Befehl [TapiListen\(\)](#) benötigt. Dem Befehl wird der Deskriptor eines Gerätetreibers und eines Fensters übergeben. Kommt auf dem Gerätetreiber ein Gespräch an, wird das Ereignis [EvtTapi](#) des Fensters ausgelöst. Die Überwachung bleibt erhalten, bis sie durch den Befehl [TapiListen\(\)](#) wieder deaktiviert wird.

Damit mit einer aktiven Verbindung weitere Aktionen (Vermitteln, Konferenzschaltungen usw.) durchgeführt werden können, werden Besitzrechte an dieser Verbindung benötigt. Diese Besitzrechte werden über die Eigenschaft [TapiFlags](#) gesteuert. Standardmäßig werden Besitzrechte angefordert. Sollen Verbindungen über ein TAPI-Gerät nur überwacht werden (Monitoring), muss das Flag [TapiListenMonitor](#) eingetragen werden. Das nächsten [TapiListen\(\)](#) wird dann ohne Besitzrechte an den Verbindungen durchgeführt.

Kontakt

Folgende Befehle können nicht auf eine CallID durchgeführt werden, für die keine Besitzrechte vorhanden sind:

- TapiCall() mit TapiCallOpAnswer, TapiCallOpDrop oder TapiCallOpHold
- TapiForward()
- TapiConference(), TapiConferenceDial(), TapiConferenceCommit()

In einem solchen Fall geben die Funktionen den Fehlerwert ErrTapiNotOwner zurück.

Die TAPI-Schnittstelle muss mindestens die Version 2.0 und der Gerätetreiber die Version 1.4 besitzen, um von CONZEPT 16 angesprochen zu werden. Die TAPI-Version des Gerätetreibers kann über die Systemeigenschaft Version des Gerätetreibers ermittelt werden.

Kontakt

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

Ausführen von Anrufoperationen

obj Call-ID

Operation

TapiCallOpAnswer

Gespräch

annehmen

Gespräch trennen

Gespräch halten

Ursprung des

Gesprächs

ermitteln

Gesprächsstatus

ermitteln

Gespräch wieder

aufnehmen

TapiCallOpSetOwner Applikation wird

Eigentümer des

Anrufs

TapiCallOpIsOwner

Überprüfung der

Eigentümerschaft

int1 wird nicht ausgewertet

Resutat int Fehlerwert oder Gesprächsstatus

Siehe Verwandte Befehle, EvtTapi

Der Befehl dient zum Durchführen bestimmter Anruf-Operationen. In (obj) wird die Call-ID übergeben, die dem Ereignis EvtTapi übergeben wurde. In (int1) kann eine der folgenden Operationen übergeben werden:

TapiCallOpAnswer Gespräch annehmen

TapiCallOpDrop Gespräch trennen

TapiCallOpHold Gespräch halten

TapiCallOpOrigin Ursprung des Gesprächs ermitteln

TapiCallOpState Gesprächsstatus ermitteln

TapiCallOpUnhold Gespräch wieder aufnehmen

TapiCallOpSetOwner Applikation wird Eigentümer des Anrufs

TapiCallOpIsOwner Überprüfung der Eigentümerschaft

Nicht alle Operationen können in jedem Zustand des Gesprächs ausgeführt werden.

Auch stellen nicht alle TAPI Service Provider alle Funktionen zur Verfügung.

Der Parameter (int2) wird zur Zeit noch nicht ausgewertet.

Als Rückgabewerte können neben ErrOk die im Abschnitt Konstanten für Fehler von der TAPI-Schnittstelle beschriebenen Konstanten zurückgegeben werden. Wird der Gesprächsstatus bzw. der Ursprung eines Gesprächs ermittelt, kann das Ergebnis mit einer der TapiCallState...- bzw. TapiCallOrigin...- Konstanten verglichen werden.



Die Funktion wird asynchron ausgeführt. Dies bedeutet, dass nicht gewartet wird, bis der jeweils gewünschte Zustand erreicht wird. Zum Beispiel wird bei

Kontakt

Aufruf von TapiCall() mit TapiCallOpAnswer nicht gewartet, bis sich der Anruf im Zustand TapiCallStateConnected befindet. Der Aufruf von TapiCall() hat nachfolgende EvtTapi-Aufrufe zur Folge.

Kontakt

obj -> TapiClose() |  | TAPI-

Gerätetreiberliste schließen

Liste der
obj

Geräte-Treiber

Verwandte

Siehe Befehle,

TapiOpen()

Der Befehl schließt die TAPI-Schnittstelle und löscht die Liste der TAPI-Geräte. In (obj) muss der Deskriptor übergeben werden, der von TapiOpen() zurückgegeben wurde.



Beispiel siehe TapiOpen().

Kontakt

obj ->

TapiConference() : 

int

Konferenz initiieren

obj Call-ID
 ConsultCall ID oder

Resultat int Fehlerwert

Siehe [TapiConferenceDial\(\)](#),

[TapiConferenceCommit\(\)](#)

Dieser Befehl initiiert eine Konferenz. In (obj) wird die Call-ID eines bereits verbundenen Anrufs ([TapiCallStateConnected](#)) angegeben werden. Dabei kann es sich sowohl um einen eingehenden als auch einen mit [TapiDial\(\)](#) gewählten, ausgehenden Anruf handeln.

Bei der erfolgreichen Durchführung liefert die Funktion eine neue Call-ID für den sogenannten Consultant-Call zurück. Diese wird dazu verwendet, die Konferenz aufzubauen. Der durch Call-ID gekennzeichnete Anruf wechselt in den Status [TapiCallStateOnHoldPendTransfer](#), er wird also auf "Halten" gelegt.

Zudem können die im Abschnitt [Konstanten für Fehler von der TAPI-Schnittstelle](#) beschriebenen Konstanten zurückgegeben werden.

Kontakt

obj ->

TapiConferenceCommit(handle1)



: int

Konferenz schalten

obj Call-ID

handle1 ConsultCall-ID

Resultat int Konferenz-Call-ID

oder Fehlerwert

Verwandte Befehle,

Siehe TapiConference(),

TapiConferenceDial()

Dieser Befehl schaltet eine Konferenz. In (obj) wird hierzu die bei TapiConference() angegebene Call-ID übergeben. In (handle1) wird die von TapiConference() zurückgelieferte ConsultCall-ID übergeben.

Der durch (obj) gekennzeichnete Call muss sich im Zustand TapiCallStateOnHold oder TapiCallStateOnHoldPendingTransfer befinden.

Der durch (handle1) gekennzeichnete Call muss sich im Zustand

TapiCallStateConnected, TapiCallStateBusy, TapiCallStateRingBack oder TapiCallStateProceeding befinden.

Ist der Befehl erfolgreich, wird eine Konferenz-Call-ID zurückgeliefert. Die verbundenen Gesprächspartner wechseln in den Status TapiCallStateConferenced.

Zudem können die im Abschnitt Konstanten für Fehler von der TAPI-Schnittstelle beschriebenen Konstanten zurückgegeben werden.

Kontakt

obj -> TapiConferenceDial(alpha1[, int2,
[int3]]) : int



Dritten Konferenzteilnehmer anrufen

obj ConsultCall-ID

alpha1 Wählstring

Optionen (optional)

0 Keine Option
TapiAsyncDial Asynchrones

int2 Wählen

TapiRawDial Wählstring
unverändert
benutzen

Timeout beim synchronen

int3 Wählen (optional)

Resultat int Fehlerwert

Verwandte Befehle,

Siehe TapiConference(),

TapiConferenceCommit()

Dieser Befehl stellt die Verbindung zu einem weiteren Gesprächspartner einer durch TapiConference() initiierten Konferenz her. In (obj) wird die ConsultCall-ID übergeben, die zuvor von TapiConference() zurückgeliefert wurde. Die zu wählende Nummer wird in (alpha1) angegeben.

Über den Parameter (int2) können zusätzliche Optionen definiert werden.

Standardmäßig wird synchron gewählt. Dabei kann in (int3) ein Timeout angegeben werden. Ist kein Timeout angegeben, wird bis zu 60 Sekunden gewartet. Mit

TapiAsyncDial in (int2) wird asynchron gewählt. Beim asynchronen Wählen werden Angaben in (int3) ignoriert. Wird in (int2) TapiRawDial angegeben, wird der Wählstring durch den Tapi-Treiber nicht für das Tapi-Device aufbereitet, sondern unverändert zum Wählen benutzt.

Beim asynchronen Wählen kann über das Ereignis EvtTapi festgestellt werden, ob eine Verbindung zustande gekommen ist oder nicht.

Als Rückgabewerte können neben ErrOk die im Abschnitt Konstanten für Fehler von der TAPI-Schnittstelle beschriebenen Konstanten zurückgegeben werden.

 Der Befehl TapiCall() mit der Option TapiCallOpDrop kann dazu verwendet werden, um den Vorgang abzubrechen. Der Status des originären Gesprächspartners wechselt dann wieder in den Status TapiCallStateConnected. Der durch ConsultCall-ID gekennzeichnete Anruf wechselt in den Status TapiCallStateIdle.

Mögliche Laufzeitfehler:

ErrValueInvalid In (int3) wurde ein anderer Wert als 0 oder eine ungültige Konstante angegeben.

obj ->
TapiDial(alpha1[, int2[, int3]]) : int

Nummer wählen

obj alpha1 int2 int3 Resultat int Siehe	Deskriptor des TAPI-Gerätetreibers Wählstring Optionen (optional) 0 <u>TapiAsyncDial</u> <u>TapiReturnCallID</u> <u>TapiRawDial</u> TimeOut beim synchronen Wählen in Millisekunden (optional) <u>ErrOk</u> , resultierende Call-ID oder Fehlerwert <u>Verwandte Befehle</u> , <u>TapiOpen()</u> , <u>Blog</u>
--	--

Wählen mit Rückgabe der Call-ID Wählstring unverändert benutzen

Keine Option Asynchrones

Wählen

Asynchrones

Wählen mit Rückgabe der Call-ID Wählstring unverändert benutzen

Wählen mit Rückgabe der Call-ID Wählstring unverändert benutzen



Mit diesem Befehl kann über ein TAPI-Gerät eine Nummer gewählt werden. In (obj) muss der Deskriptor des TAPI-Gerätes, der zuvor mit dem Befehl CteRead() aus der Liste der TAPI-Geräte ermittelt wurde, übergeben werden. Die Zeichenkette mit der zu wählenden Nummer wird in (alpha1) übergeben.

Das Format und die zulässigen Zeichen in diesem Wählstring sind von dem verwendeten Gerätetreiber abhängig.

Über den Parameter (int2) können zusätzliche Optionen definiert werden.

Standardmäßig wird synchron gewählt. Dabei kann in (int3) ein Timeout angegeben werden. Ist kein Timeout angegeben, wird bis zu 60 Sekunden gewartet. Mit

TapiAsyncDial in (int2) wird asynchron gewählt. Beim asynchronen Wählen werden Angaben in (int3) ignoriert. Mit der Option **TapiReturnCallID** wird zusätzlich zum asynchronen Wählen die resultierende Call-ID zurückgegeben. Wird in (int2) **TapiRawDial** angegeben, wird der Wählstring durch den Tapi-Treiber nicht für das Tapi-Device aufbereitet, sondern unverändert zum Wählen benutzt.

Beim asynchronen Wählen kann über das Ereignis EvtTapi festgestellt werden, ob eine Verbindung zustande gekommen ist oder nicht. Die Abfrage des Verbindungsfehlers kann über den Aufruf **TapiCall(..., TapiCallOpDisconnectMode)** erfolgen.

Als Rückgabewerte können neben **ErrOk** die im Abschnitt Konstanten für Fehler von der TAPI-Schnittstelle beschriebenen Konstanten zurückgegeben werden. Bei der Option **TapiReturnCallID** ist das Resultat im erfolgreichen Fall die resultierende

Kontakt

Call-ID.

Bei der Programmierung sollte ein asynchroner Dialog verwendet werden. Der Befehl kann nicht abgebrochen werden und bis zu Ende des Timeouts dauern, bis er vom TAPI-Treiber zurückkehrt.

Wird ein TapiDial() durchgeführt und anschliessend ein TapiDial() mit einem anderen TAPI-Gerät durchgeführt, wird das aktuelle TAPI-Gerät geschlossen.

Mögliche Laufzeitfehler:

ErrValueInvalid In (int2) wurde ein anderer Wert als 0 oder eine ungültige Konstante angegeben.

obj ->
TapiForward(alpha1[,]
int2, [int3]) : int
Anruf weiterleiten
obj Call-ID
alpha1 Wählstring
Optionen (optional)
0 Keine Option
TapiAsyncDial Asynchrones
int2 Wählen
TapiRawDial Wählstring
unverändert
benutzen
Timeout beim
int3 synchronen Wählen
(optional)
Resultat int Fehlerwert
Verwandte Befehle,
Siehe
TapiDial()

Dieser Befehl leitet einen eingegangenen Anruf (obj) an die angegebene Adresse (alpha1) weiter. Die Anruf-Parteien müssen bereits verbunden sein (TapiCallStateConnected), damit die Weiterleitung funktioniert.

Über den Parameter (int2) können zusätzliche Optionen definiert werden.

Standardmäßig wird synchron gewählt. Dabei kann in (int3) ein Timeout angegeben werden. Ist kein Timeout angegeben, wird bis zu 60 Sekunden gewartet. Mit TapiAsyncDial in (int2) wird asynchron gewählt. Beim asynchronen Wählen werden Angaben in (int3) ignoriert. Wird in (int2) TapiRawDial angegeben, wird der Wählstring durch den Tapi-Treiber nicht für das Tapi-Device aufbereitet, sondern unverändert zum Wählen benutzt.

Der Befehl liefert ErrOk, wenn die Weiterleitung bei asynchroner Verarbeitung erfolgreich eingeleitet wurde, beziehungsweise wenn bei der synchronen Verarbeitung eine Verbindung hergestellt wurde. Zudem können die im Abschnitt Konstanten für Fehler von der TAPI-Schnittstelle beschriebenen Konstanten zurückgegeben werden.

Mögliche Laufzeitfehler:

ErrValueInvalid In (int3) wurde ein anderer Wert als 0 oder eine ungültige Konstante angegeben.

Kontakt

obj -> TapiInfo(int1, var var2) : int | 

Informationen zu einem Anruf ermitteln

Call-ID oder Deskriptor des
obj TAPI-Gerätetreibers

Informationstyp
TapiInfoCallerID

Nummer des

Anrufers
Name des Anrufers

TapiInfoCalledID

Nummer des
Angerufenen

TapiInfoCalledIDName

Name des

int1 TapiInfoConnectedID

Angerufenen
Nummer des
verbundenen
Teilnehmers

TapiInfoConnectedIDName Name des
verbundenen

Teilnehmers
Statusinformationen
des TAPI-Geräts

var2 angeforderte Information

Resultat int Fehlerwert

Siehe Verwandte Befehle

Mit diesem Befehl werden Informationen zu einem TAPI-Objekt abgefragt. In (obj) wird die Call-ID des Anrufs übergeben. Die Call-ID wird bei dem Ereignis EvtTapi vom System übergeben. Das Ergebnis wird in (var2) zurückgegeben und ist vom Typ alpha.

Soll der Status ermittelt werden, muss an Stelle der Call-ID der Deskriptor des TAPI-Gerätes übergeben werden. Als (var2) muss der Deskriptor einer Cte-Liste übergeben werden.

In (int1) wird der Informationstyp angegeben. Folgende Konstanten können hier übergeben werden:

- TapiInfoCallerID

Es wird die Kennung (Rufnummer) des Anrufers ermittelt. Das Ergebnis entspricht der im EvtTapi übergebenen Nummer (aCallerID).

- TapiInfoCallerIDName

Es wird der Name des Anrufers ermittelt.

- TapiInfoCalledID

Es wird die Kennung (Rufnummer) des angerufenen ermittelt. Das Ergebnis entspricht der im EvtTapi übergebenen Nummer (aCalledID).

- TapiInfoCalledIDName

Es wird der Name des Angerufenen ermittelt.

- TapiInfoConnectedID

Kontakt

Es wird die Kennung (Rufnummer) des verbundenen Teilnehmers ermittelt. Die Abfrage ist erst nach erfolgreichem Zustandekommen einer Verbindung (TapiCallStateConnected) sinnvoll.

- TapiInfoConnectedIDName

Es wird der Name des verbundenen Teilnehmers ermittelt. Die Abfrage ist erst nach erfolgreichem Zustandekommen einer Verbindung (TapiCallStateConnected) sinnvoll.

- TapiInfoStatus

Soll der Status eines mit TapiListen() überwachten Gerätes abgefragt werden, muss der Deskriptor des Tapi-Gerätes als (obj) übergeben werden. Als (var2) muss der Deskriptor einer Cte-Liste übergeben werden. Die Liste ist nach dem Aufruf um ein Cte-Item pro Anwendung, die das gleiche TAPI-Gerät geöffnet hat, erweitert.

Die in der Liste enthaltenen CteItem-Objekte haben die folgenden Eigenschaften:

- ◆ Name - Name der Anwendung, der z. B. zur Anzeige verwendet werden kann. Für CONZEPT 16 beinhaltet die Eigenschaft "CONZEPT 16 TAPI".
- ◆ Custom - Nicht vorbelegt. Die Eigenschaft kann verwendet werden.
- ◆ ID - Nicht vorbelegt. Die Eigenschaft kann verwendet werden.
- ◆ TapiNameComputer - Rechnername, auf dem die Anwendung läuft.
- ◆ TapiUserName - Name des Benutzers in dessen Kontext die Anwendung läuft.
- ◆ TapiModuleFilename - Name des Moduls der Anwendung (bei CONZEPT 16 "c16_objw.dll").
- ◆ TapiOwnerRequest - Gibt an, ob versucht wurde das TAPI-Gerät mit Besitzrechten zu öffnen (true) oder nicht (false).

Die Eigenschaften können nur gelesen werden, lediglich die Eigenschaften Name, Custom und ID können gelesen und geschrieben werden.

Wird der Typ (HdlType) bzw. der Subtyp (HdlSubType) des Deskriptors der Listen-Items mit HdlInfo() ermittelt, wird HdlTapi bzw. TapiHdlStatusItem zurückgegeben.

Beispiel:

```
tDataListTapiApps # $dlTapiApps;tStatusList # CteOpen(_CteList);tError # Device->TapiInfo(_TapiIn
```



Ob die Informationen ermittelt werden können, hängt davon ab, ob der TAPI-Treiber die Informationen zu Verfügung stellt. Hat der TAPI-Treiber die jeweiligen Informationen, dann ist zu beachten, dass diese vom Zustand des Anrufs abhängen. TapiInfoConnectedID und TapiInfoConnectedIDName sind z. B. nur im Zustand TapiCallStateConnected vorhanden.

Der Rückgabewert ist ein Fehlerwert (ErrTapi...) oder ErrOk, wenn der Aufruf erfolgreich war. In diesem Fall kann (var2) jedoch trotzdem leer sein.

Mögliche Laufzeitfehler:

Kontakt

ErrFldType In (var2) wird der falsche VariablenTyp übergeben.

ErrValueInvalid In (int1) wurde ein falscher Informationstyp angegeben.

Bei dem übergebenen Deskriptor (bei **TapiInfoStatus**) handelt es

ErrHdlInvalid sich nicht um ein Tapi-Gerät oder es ist kein **TapiListen()** auf dem
Tapi-Gerät aktiv.

Kontakt

**obj -> TapiListen(logic1,
handle2) : int**



Eingehende Anrufe überwachen
Deskriptor des
obj

Gerätetreibers
Überwachung
logic1 aktivieren /
deaktivieren
Deskriptor des
handle2 Frame-Objektes,
das das Ereignis
erhält

Resultat int Fehlerwert

Verwandte
Siehe [Befehle, EvtTapi,](#)
[TapiOpen\(\), Blog](#)

Mit diesem Befehl ist ein Überwachen von eingehenden Anrufen über die Tapischnittstelle möglich, sofern die TAPI-Geräte und deren Treiber dies unterstützen (Voraussetzung ist unter anderem TAPI Version 2.0). Sollen mehrere Leitungen überwacht werden, kann der Befehl auch mehrfach ausgeführt werden.

Als (obj) wird der Deskriptor des zu überwachenden Gerätetreibers übergeben.

Dieser kann zuvor aus der Liste der Gerätetreiber ermittelt werden (vgl. [TapiOpen\(\)](#)).

Der Parameter (logic1) schaltet die Überwachung ein (true) oder aus (false). In (handle2) wird der Deskriptor eines Frames angegeben, bei dem das Ereignis [EvtTapi](#) ausgelöst wird, wenn ein Anruf auf diesem Gerät eingeht.

Als Rückgabewerte können neben [ErrOk](#) die im Abschnitt [Konstanten für Fehler von der TAPI-Schnittstelle](#) beschriebenen Konstanten zurückgegeben werden.

Mögliche Laufzeitfehler:

[ErrHdlInvalid](#) Leitung bereits überwacht oder übergebener Frame nicht vorhanden

Kontakt

TapiOpen() : handle TAPI-



Gerätetreiber ermitteln

Resultat handle Deskriptor einer

Cte-Liste

Cte-Befehle, Verwandte Befehle,

Siehe

TapiClose()



Mit diesem Befehl werden die aktuellen TAPI-Geräte (TapiDevice) ermittelt. Als Resultat wird ein Deskriptor einer Cte-Liste zurückgeliefert.

Der Zugriff auf das TapiDevice-Objekt kann mit Hilfe der normalen Cte-Befehle erfolgen. Die Elemente der Liste entsprechen den TAPI-Geräten. Das Objekt TapiDevice enthält die Eigenschaften Name und Version.

Beispiele:

```
// Name des ersten TAPI-Device ermittelntTapiHdl # TapiOpen();if (tTapiHdl > 0){ tTapiDevice # t
```

Kontakt

obj ->

TapiPickup(alpha1) |      

: int

Anruf heranzuholen

 Deskriptor des
obj

TAPI-Gerätetreibers

alpha1 Zielapparat
 CallID oder

Resultat **int Fehlerwert**

Siehe Verwandte Befehle,

TapiDial()

Der Befehl TapiPickup kann dazu verwendet werden, einen Anruf heranzuholen, der auf einem anderen Apparat ankommt.

In (obj) muss der Deskriptor eines **TAPI-Gerätetreibers** übergeben werden. Der Gerätetreiber muss mit TapiListen() überwacht werden. Im Argument (alpha1) wird die Nummer des Zielapparates angegeben, von dem der eingehende Anruf herangeholt werden soll.

Wurde der Befehl erfolgreich ausgeführt, wird die CallID des übernommenen Anrufs zurückgegeben. Folgende Fehlerwerte sind definiert:

- ErrTapiNoListen

Das angegebene TapiDevice (obj) wird nicht durch TapiListen() überwacht.

- ErrTapiUnavail

Die Pickup-Funktionalität wird durch den Gerätetreiber nicht unterstützt oder ist zum Zeitpunkt des Aufrufes nicht möglich.

- ErrTapiDialString

Ungültige Zieladresse (alpha1) angegeben.

- ErrTapiFailed

Der Vorgang ist fehlgeschlagen.

Mögliche Laufzeitfehler:

ErrHdlInvalid Bei (obj) handelt es sich nicht um ein **TAPI-Gerätetreiber**

Kontakt

Web-Schnittstellen-Befehle

Befehle zur Programmierung einer Web-Anwendung Beispiel,

Verwandte

Befehle, Liste

Siehe sortiert nach

Gruppen,

Alphabetische

Liste aller

Befehle

Die Wse-Befehle funktionieren nur im Zusammenhang mit der Web-Schnittstelle. Prozeduren, die von der Web-Schnittstelle aufgerufen werden, können nur Befehle enthalten, die mit  gekennzeichnet sind.

Befehle

- WseArg
- WseInfo
- WseReturn
- WseStatus
- WseStrCnv

Konstanten

- WseArgCall
- WseArgName
- WseArgReq
- WseCnvArg
- WseCnvCtrl
- WseCnvHTML
- WseCnvISO
- WseInfoAppID
- WseInfoAppName
- WseInfoAppURL
- WseInfoErrorHandler
- WseInfoHTTP
- WseInfoLogPath
- WseInfoModulePath
- WseInfoReqData
- WseInfoReqMethod
- WseInfoReqPath
- WseInfoReqProtocol
- WseInfoRootPath
- WseInfoUserID
- WseInfoUserIP
- WseInfoUserNumber
- WseInfoUserSessionTime
- WseNoParse
- WseRetExpires
- WseRetFile
- WseRetHeader

Kontakt

- [WseRetString](#)
- [WseRetText](#)
- [WseSendAfter](#)
- [WseTerm](#)
- [WseUserHTML](#)
- [WseUserInit](#)
- [WseUserProc](#)
- [WseUserTerm](#)



**WseArg(int1, alpha2[,
int3]) : alpha**

Argumente ermitteln

Parametertyp

WseArgName Parametername

int1	<u>WseArgReq</u>	ermitteln Parameterwert
	<u>WseArgCall</u>	ermitteln Übergabeparameter

alpha2 Argumentname

int3 Argumentposition

Resultat alpha Name oder Argument

Siehe Verwandte Befehle

Informationen von einer HTML-Seite können auf unterschiedlichen Wegen in die Funktionen der Web-Anwendung transportiert werden. Wird innerhalb einer HTML-Seite ein Link angeklickt, der zur Web-Schnittstelle führt, oder ein Formular abgeschickt, wird die Request-Prozedur ausgeführt. Innerhalb der Request-Prozedur kann auf den Query-String oder den Response-Body des Requests zugegriffen werden.

Einem Link können Parameter übergeben werden. Die Parameter werden von der eigentlichen URL durch ein "?" getrennt.

http://www.vectorsoft.de/scripts/c16_web.dll?page1=next

In diesem Fall wird page1=next als Query-String übermittelt. Der Name des ersten Parameters (page1) kann mit dem Aufruf **WseArg(WseArgName, '')** ermittelt werden. Ist der Name des Parameters bekannt, kann der Wert (next) mit **WseArg(WseArgReq, 'page1')** abgefragt werden.

Die Anzahl der Parameter ist nur über die Größe des Query-Strings beschränkt. Der Query-String darf 4 KB nicht überschreiten. In manchen Fällen kommt es allerdings schon bei Query-Strings die 1 KB überschreiten zu Problemen.

Werden mehrere Parameter übergeben, kann ebenfalls über die Reihenfolge der Parameter zugegriffen werden. Die Position des Parameters wird dabei in (int3) angegeben. Der Befehl **WseArg(WseArgName, '', 2)** ermittelt den Namen des zweiten Parameters. Die Namen der Parameter können in (alpha2) eingeschränkt werden. Wird in (alpha2) 'p*' angegeben, wird die Auswahl der Parameter, auf die Parameter eingeschränkt, dessen Namen mit "p" beginnen. In dem Beispiel wird der Name des zweiten Parameters der mit "p" beginnt ermittelt. Spielt der Name keine Rolle, kann mit dem Befehl **WseArg(WseArgReq, '', 2)** auch direkt auf den Wert zugegriffen werden.



Wird in der Konfiguration der Web-Schnittstelle festgelegt, dass die Benutzer-ID im Query-String (**web_uid mode=query**) übertragen wird, ist der erste Parameter immer C16UID und der Wert die Benutzer-ID des Web-Benutzers.

Formulardaten werden auf die gleiche Weise übertragen. Innerhalb eines Formulars kann die Methode der Übertragung festgelegt werden. Dazu wird innerhalb des

Kontakt

<form>-Tags method=get oder method=post angegeben. Die Formulardaten werden dann innerhalb des Query-Strings oder des Response-Bodys übertragen. Zum Ermitteln der Formulardaten spielt die Übertragungsmethode keine Rolle.



Gerade bei größeren Formularen sind die kritischen 1 KB schnell erreicht. Formulare sollten mit der Methode POST übertragen werden. Zudem werden so die Formulardaten nicht im Adressfeld des Browsers angezeigt.

Die Namen der Formularfelder und deren Inhalte können auf die gleiche Weise wie die Parameter eines Links ermittelt werden.

Innerhalb einer HTML-Seite können Funktionen aus der Applikation aufgerufen werden. Diese Aufrufe erfolgen mit der Anweisung C16.CALL(). Der aufgerufenen Funktion können Parameter übergeben werden. Diese Parameter können mit der Anweisung WseArg(WseArgCall, ...) ermittelt werden. Da die Parameter keine Namen besitzen, muss die Abfrage über die Reihenfolge der Parameter erfolgen. (alpha2) ist in diesem Fall immer ''.

Zur Ermittlung der Daten stehen verschiedene Wege zur Verfügung:

- Ermittlung der Werte über die Position

Dazu wird in (int1) WseArgReq (Aufruf aus einem Request) oder WseArgCall (Aufruf über C16.CALL()), in (alpha2) '' und in (int3) die Position des Wertes übergeben.

Existiert zu der übergebenen Position kein Wert wird ein Leerstring zurückgegeben. Bei der Ermittlung über die Position besteht keine Möglichkeit zwischen den Fällen "Wert ist leer" und "Wert ist nicht vorhanden" zu unterscheiden. In beiden Fällen wird eine leere Zeichenkette zurückgegeben.

```
// Ermittlung aller Parameter oder bis zum ersten leeren Parameter tCounter # 1;aArgValue #
```

- Ermittlung des Wertes über den Namen

Dazu wird zunächst der Name des Parameters ermittelt. Dies erfolgt mit dem Parameter WseArgName in (int1). Die Namen können ähnlich wie die Werte über die Position ((alpha2) = "", (int3) = Position) ermittelt werden.

Existiert zu einer Position kein Name, wird ein Leerstring zurückgegeben.

```
// Ermittlung aller Parameter tCounter # 1;aArgName # WseArg(_WseArgName, '', tCounter);whi
```

Zusätzlich kann in (alpha2) noch ein Suchwert mit angegeben werden. Es werden dann nur die Namen ermittelt, die dem Suchbegriff ähnlich (Operator =*) sind. Nachdem der Name ermittelt ist, kann dessen Wert mit WseArg(WseArgReq, <Name>) ermittelt werden. Bei diesem Verfahren kann unterschieden werden, ob ein Name nicht vorhanden (der Name wird nicht ermittelt) oder der Wert leer ist (der Name wird ermittelt, als Wert wird ein Leerstring zurückgegeben).

- Ermittlung von Funktionsargumenten

Kontakt

Wird eine Funktion über das Kommando **C16.CALL()** aufgerufen, können die Argumente über die Reihenfolge mit dem Typ **WseArgCall** abgefragt werden. Der Parameter (alpha2) ist dabei immer ''.

Kontakt

WseInfo(int1[, alpha2]) :



alpha

Web-Informationen ermitteln

Zu ermittelnde

int1

Information (siehe Text)

alpha2 ID des Eintrags

Resultat alpha Wert des Eintrags

Siehe Verwandte Befehle

Mit diesem Befehl können verschiedene Informationen über die Anfrage, den Web-Benutzer und den Browserrechner ermittelt werden. In (int1) wird die zu ermittelnde Information angegeben:

- **WseInfoHTTP**

Mit diesem Parameter können Informationen aus dem HTTP-Header des Requests ermittelt werden. Die entsprechende Bezeichnung muss in (alpha2) angegeben werden.

- **WseInfoUserID**

Über diesen Parameter wird die ID des Web-Benutzers ermittelt. Der Web-Benutzer unterscheidet sich vom Datenbank-Benutzer. Da mehrere Web-Benutzer sich eine Datenbankverbindung teilen können, arbeiten mehrere Web-Benutzer der Applikation mit der gleichen Datenbankbenutzer-ID in der Datenbank. Eine Unterscheidung kann nur über die ID oder die Nummer des Web-Benutzers erfolgen. Ebenso ist nicht gesichert, dass ein Web-Benutzer für jeden Request die gleiche Verbindung zur Datenbank erhält. Ein Web-Benutzer arbeitet also auch mit unterschiedlichen Datenbank-IDs. Da die Benutzer-ID des Web-Benutzers eine 24stellige Zeichenkette ist, eignet sie sich nicht als Bestandteil von Namen von temporären (sessionbezogenen) Texten oder Selektionen. Zu diesem Zweck sollte die Benutzer-Nummer (**WseInfoUserNumber**) verwendet werden.

- **WseInfoUserNumber**

Dieser Parameter ermittelt die eindeutige Nummer des Web-Benutzers innerhalb der Applikation.

Im Unterschied zur Benutzer-ID wird hier eine Zahl zurückgegeben.

- **WseInfoUserIP**

Dies ist die IP-Adresse, die als Quelle des Requests vom Client ermittelt wurde.

Bei Verwendung von Proxy-Servern oder Gateways muss diese Adresse nicht der des Browserrechners entsprechen. Eine Unterscheidung von Anwender-Rechnern anhand der IP-Adresse ist daher nicht immer möglich. Unter Umständen (zum Beispiel bei "reverse hosting") haben alle Requests dieselbe IP-Adresse.

- **WseInfoUserSessionTime**

Es wird die Zeit zurückgegeben, die seit der Anmeldung des Web-Benutzers vergangen ist. Der Befehl liefert dabei die Anzahl der Sekunden zurück. Der

Kontakt

Wert wird als Zeichenkette zurückgegeben, muss also zur weiteren Verarbeitung gewandelt werden.

- WseInfoReqProtocol

Bei diesem Parameter wird das verwendete Protokoll (HTTP/1.0 oder HTTP/1.1) zurückgegeben.

- WseInfoReqMethod

Die Request-Methode wird zurückgegeben. Für die weitere Verarbeitung im Client ist nur die Unterscheidung zwischen einem GET- und einem POST-Request nötig, andere Request-Methoden werden im Client behandelt.

- WseInfoReqData

Abhängig von der verwendeten Request-Methode wird der Request-String (Methode GET) oder der Request-Body (Methode POST) zurückgegeben. Die Länge der Daten darf dabei 4 KB nicht überschreiten.

- WseInfoReqPath

Der Request-Pfad zwischen der Schnittstelle (c16_web.dll) und dem ersten Parameter wird zurückgegeben.

Die folgenden Parameter ermitteln entsprechende Einträge in der Konfigurationsdatei:

- WseInfoAppID

Dieser Parameter ermittelt den Eintrag von web_app_id in der Konfigurationsdatei der Web-Schnittstelle.

- WseInfoAppName

Ermittelt den Applikationsnamen. Der Name steht in der Konfigurationsdatei der Web-Schnittstelle vor den Angaben der Applikation in eckigen Klammern [...].

- WseInfoAppURL

Ermittelt den Eintrag von web_url_id in der Konfigurationsdatei der Web-Schnittstelle.

- WseInfoErrorPath

Ermittelt den Eintrag von web_error_path in der Konfigurationsdatei der Web-Schnittstelle.

- WseInfoLogPath

Ermittelt den Eintrag von web_log_path in der Konfigurationsdatei der Web-Schnittstelle.

- WseInfoModulePath

Ermittelt den Eintrag von web_module_path in der Konfigurationsdatei der Web-Schnittstelle.

- WseInfoRootPath

Kontakt

Ermittelt den Eintrag von web_root_path in der Konfigurationsdatei der Web-Schnittstelle.

WseReturn(int1, int2, alpha3, var4[, int5])

: int



Seite oder Seiteninhalt zurückgeben

Art des Rückgabewertes

WseRetHeader Es wird nur der
HTTP-Header

WseRetString übertragen
In (var4) wird
eine Zeichenkette

int1

WseRetText angegeben
Ein bestehender
Text wird

WseRetFile übertragen
Eine externe
Datei wird
übergeben

WseRetExpires Verfallsdatum für
die übertragenen
Daten definieren

int2

Optionen zur Verarbeitung der

Rückgabedaten

WseNoParse Die
zurückgegebenen
Daten werden
nicht durch den
Client
interpretiert

WseSendAfter Die
zurückgegebenen
Daten werden erst
nach dem
Beenden der
Prozedur an den
Web-Server

WseTerm übergeben
Die Session wird
nach der

WseCnvISO Rückgabe beendet
Der Rückgabewert
wird in den
Zeichensatz ISO
8859-1 (Latin-1)

WseCnvHTML umgewandelt
Sonderzeichen

WseCnvCtrl werden gewandelt
Die Steuerzeichen
im HTML-Text
werden

umgewandelt

alpha3	MIME-Typ
	HTTP-Header, Zeichenkette,
var4	Textpuffer, externe Datei oder
	Verfallsdatum
int5	HTTP-Statuscode (optional)
Resultat int	Fehlerwert
Siehe	<u>Verwandte Befehle</u>

Über diesen Befehl können Informationen vom Client an den Web-Server zurückgegeben werden. Bei diesen Informationen kann es sich um HTML-Text handeln, der in eine bestehende Seite integriert wird, um eine ganze HTML-Seite, die an den Browser übermittelt werden soll oder um eine Datei beliebigen Typs, die zum Browserrechner übertragen werden soll. Zusätzlich kann für die übertragenen Daten auch ein Verfallsdatum beim Client definiert werden. Die Daten, die an den Web-Server und damit an den Browser übertragen werden, können unterschiedlich vorliegen.

Die Art der Daten wird in (int1) bestimmt:

- WseRetHeader

Mit dieser Option kann der HTTP-Header um zusätzliche Felder ergänzt werden.

- WseRetString

In (var4) wird eine Zeichenkette angegeben.

- WseRetText

Ein bestehender Text wird übertragen. Der Text liegt in einem Textpuffer vor. Der Textpuffer wird in (var4) übergeben.

- WseRetFile

Eine externe Datei wird übergeben. In (var4) steht der Pfad und der Name der Datei.

- WseRetExpires

Das Verfallsdatum für die übertragenen Daten wird festgelegt. Bis dieses Datum abgelaufen ist, wird der Client keine neue Anfrage für die gleichen Daten senden und die bereits übertragenen Daten aus seinem Cache verwenden. Das Ablaufdatum wird als Wert vom Typ caltime in (var4) definiert, (int2) und (alpha3) werden dann nicht verwendet. Wird über WseReturn() kein Verfallsdatum definiert, wird automatisch ein bereits abgelaufenes für die Daten verwendet.



Voraussetzung für die korrekte Funktionsweise des Verfallsdatums ist ein Client, der das Verfallsdatum von übertragenen Daten berücksichtigt.

In (int2) können besondere Optionen zur Verarbeitung der Rückgabedaten gesetzt werden:

Kontakt

- WseNoParse

Die zurückgegebenen Daten werden nicht durch den Client interpretiert. Der Web-Benutzer kann sofort einen weiteren Request verarbeiten. Diese Option ist automatisch aktiv, wenn der MIME-Typ nicht text/html ist.

- WseSendAfter

Die zurückgegebenen Daten werden erst nach dem Beenden der Prozedur an den Web-Server übergeben. Sollte ein interner Text übergeben werden, ist darauf zu achten, dass der entsprechende Puffer nach Prozedurende noch vorhanden ist. Diese Art der Rückgabe ist etwas schneller, wenn WseNoParse angegeben wird.

- WseTerm

Die Session wird nach der Rückgabe beendet, der Web-Benutzer wird entfernt und die Benutzer-ID ist anschliessend nicht mehr gültig.

- WseCnvISO

Der Rückgabewert wird in den Zeichensatz ISO 8859-1 (Latin-1) umgewandelt. Dies wird notwendig, wenn Sonderzeichen (Umlaute etc.) in dem Rückgabewert vorkommen.

- WseCnvHTML

Sonderzeichen werden in &#<Nummer>; gewandelt.

- WseCnvCtrl

Die Steuerzeichen im HTML-Text werden umgewandelt.

In (alpha3) wird die Art der zurückgegebenen Daten übergeben. Im Falle eines HTML-Textes wird hier der MIME-Typ text/html angegeben. Es können aber auch andere Daten übertragen werden. Die wichtigsten MIME-Typen sind in der nachfolgenden Liste zusammengefasst:

text/html	HTML-Text
text/xml	XML-Text
application/octet-stream	Binäres Format
application/postscript	Postscript-Datei
application/msword	Microsoft Word Dokument
application/pdf	Adobe Acrobat Dokument
application/zip	komprimiertes Archiv
application/rtf	Rich-Text-Dokument
image/gif	GIF-Datei
image/jpeg	JPG-Datei
image/tiff	TIF-Datei
audio/mpeg	MPEG-Datei (nur Ton)
video/mpeg	MPEG-Datei

Eine vollständige Liste registratorer MIME-Typen kann über die Homepage der IANA (<http://www.iana.org/>) abgerufen werden. Nicht standardisierte MIME-Subtypen

Kontakt

werden mit einem vorangestellten x- angegeben (audio/x-wav).

Die Datenquelle wird im Parameter (var4) angegeben. Je nach Art des Rückgabewertes (int1) müssen hier unterschiedliche Angaben gemacht werden:

int1 var4

WseRetHeader Felder und Werte des HTTP-Response-Headers

WseRetString Zeichenkette

WseRetText Deskriptor des Textpuffers

WseRetFile Pfad und Dateiname

WseRetExpires Verfallsdatum der übertragenen Daten vom Typ caltime

Im Parameter (int5) kann ein Statuscode übergeben werden. Dieser Statuscode entspricht einem HTTP-Status. Eine genaue Beschreibung der HTTP-Status befindet sich im RFC 2068 (Request For Comments).

Bei Fehlerwert 0 ist kein Fehler aufgetreten, bei Fehlerwert -50061 wurde ein weiterer WseReturn()-Befehl ausgeführt, nachdem bereits ein Befehl mit der Option WseSendAfter oder mit einem anderen MIME-Typ ausgeführt wurde.

Kontakt

WseStatus() : int  Status des Web-

Benutzers ermitteln

Resultat **int** Status des Web-Benutzers

Siehe [Verwandte Befehle](#)

Mit diesem Befehl kann der Status des Web-Benutzers ermittelt werden. Der Rückgabewert kann mit folgenden Konstanten verglichen werden:

- [WseUserInit](#)

Dieser Wert wird zurückgegeben, wenn der Web-Benutzer zum ersten mal die Request-Prozedur aufruft. Dieser Zustand sollte abgefragt und gegebenenfalls globale Datenbereiche angelegt werden.

- [WseUserProc](#)

Die Request-Prozedur wurde erneut aufgerufen.

- [WseUserHTML](#)

Die Funktion wurde über den Befehl [C16.CALL\(\)](#) aus einer HTML-Seite aufgerufen.

- [WseUserTerm](#)

Die Request-Prozedur wird nochmals aufgerufen, wenn der Web-Benutzer vom Client entfernt wird. Dieser Zustand sollte abgefragt und gegebenenfalls bei [WseUserInit](#) angelegte globale Datenbereiche gelöscht werden.

Beispiel:

```
sub WebReq(){  switch (WseStatus()) {    case _WseUserInit : ...    case _WseUserTerm : ...  }}
```

Kontakt

**WseStrCnv(int1, alpha2[,
alpha3]) : alpha** Zeichenkette
konvertieren



Art der Umwandlung
_WseCnvISO Der Rückgabewert

wird in den
Zeichensatz ISO
8859-1 (Latin-1)

umgewandelt

_WseCnvHTML Sonderzeichen
werden

int1

_WseCnvCtrl umgewandelt
Steuerzeichen im
HTML-Text werden

_WseCnvArg umgewandelt
Zwei Werte zu
einem
<Name>=<Value>
Paar wandeln

alpha2 umzuwandelnder Text

alpha3 Argumentwert

Resultat **alpha** umgewandelter Text

Siehe **Verwandte Befehle**

Mit diesem Befehl wird eine bestehende Zeichenkette in ein anderes Format umgewandelt. Die Konvertierung kann auch direkt beim Befehl **WseReturn()** angegeben werden, gilt dann aber für alle zurückgegebenen Daten. Mit diesem Kommando kann ein Rückgabewert unterschiedlich umgewandelt werden.

Beispiel:

```
WseReturn(_WseRetString, 0, 'text/html',
```

```
WseStrCnv(_WseCnvISO, 'Article-Description: <b>
```

Konstanten für Web-Schnittstelle

Konstanten für die Web-Schnittstelle

Siehe [Web-Schnittstellen-Befehle](#)

- [WseArgCall](#)
- [WseArgName](#)
- [WseArgReq](#)
- [WseCnvArg](#)
- [WseCnvCtrl](#)
- [WseCnvHTML](#)
- [WseCnvISO](#)
- [WseInfoAppID](#)
- [WseInfoAppName](#)
- [WseInfoAppURL](#)
- [WseInfoErrorPath](#)
- [WseInfoHTTP](#)
- [WseInfoLogPath](#)
- [WseInfoModulePath](#)
- [WseInfoReqData](#)
- [WseInfoReqMethod](#)
- [WseInfoReqPath](#)
- [WseInfoReqProtocol](#)
- [WseInfoRootPath](#)
- [WseInfoUserID](#)
- [WseInfoUserIP](#)
- [WseInfoUserNumber](#)
- [WseInfoUserSessionTime](#)
- [WseNoParse](#)
- [WseRetExpires](#)
- [WseRetFile](#)
- [WseRetHeader](#)
- [WseRetString](#)
- [WseRetText](#)
- [WseSendAfter](#)
- [WseTerm](#)
- [WseUserHTML](#)
- [WseUserInit](#)
- [WseUserProc](#)
- [WseUserTerm](#)

_WseArgCall

Übergabeparameter ermitteln

Wert 2

Verwandte

Siehe Befehle,

WseArg()

Option bei WseArg(). Der Befehl WseArg(_WseArgCall, ", ...) ermittelt die Übergabeparameter einer Funktion, die mit C16.CALL() aus einer HTML-Seite aufgerufen wurde.

Kontakt

_WseArgName

Parametername ermitteln

Wert 1

Verwandte

Siehe Befehle,

WseArg()

Option bei WseArg(). Mit diesem Übergabeparameter wird der Name eines Parameters im Query-String oder im Response-Body ermittelt.

Kontakt

_WseArgReq

Parameterwert ermitteln

Wert 0

Verwandte

Siehe Befehle,

WseArg()

Option bei WseArg(). Mit diesem Parameter wird der Wert eines Parameters im Query-String oder im Response-Body ermittelt.

Kontakt

_WseCnvArg

zwei Werte zu einem <Name>=<Value> Paar wandeln Wert 8 /
0x08

Verwandte

Siehe Befehle,

WseStrCnv()

Option bei WseStrCnv(). Mit diesem Parameter werden zwei Werte zu einem <Name>=<Value> Paar gewandelt. In (alpha2) steht der Name und in (alpha3) der Wert.

Kontakt

_WseCnvCtrl

Steuerzeichen im HTML-Text werden umgewandelt Wert 4 /
0x04

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Folgende Zeichen werden gewandelt:

< nach < >

nach > " nach

" & nach

&

Kontakt

_WseCnvHTML

Sonderzeichen werden umgewandelt

Wert 2 / 0x02

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Sonderzeichen werden in &#<Nummer>; gewandelt.

Kontakt

_WseCnvISO

Der Rückgabewert wird umgewandelt

Wert 1 / 0x01

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Der Rückgabewert wird in den Zeichensatz ISO 8859-1 (Latin-1) umgewandelt. Dies wird notwendig, wenn Sonderzeichen (Umlaute usw.) in dem Rückgabewert vorkommen.

Kontakt

_WseInfoAppID

Ermittelt den Eintrag von web_app_id

Wert 9

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Eintrag von web_app_id in der Konfigurationsdatei der Web-Schnittstelle.

Kontakt

_WseInfoAppName

Ermittelt den Applikationsnamen

Wert 10

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Applikationsnamen.

Der Applikationsname steht in der Konfigurationsdatei der Web-Schnittstelle vor den Angaben der Applikation in eckigen Klammern ([...]).

Kontakt

_WseInfoAppURL

Ermittelt den Eintrag von web_url_id

Wert 8

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Eintrag von web_url_id in der Konfigurationsdatei der Web-Schnittstelle.

_WseInfoErrorPath

Ermittelt den Eintrag von web_error_path

Wert 13

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Eintrag von web_error_path in der Konfigurationsdatei der Web-Schnittstelle.

Kontakt

_WseInfoHTTP

Informationen aus dem HTTP-Header

Wert 0

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Mit diesem Parameter können Informationen aus dem HTTP-Header des Requests ermittelt werden.

Die entsprechende Bezeichnung muss in alpha2 angegeben werden.

Bezeichnung	Beispielrückgabe
HTTP_ACCEPT	image/gif, image/x-bitmap, image/jpeg, */*
HTTP_ACCEPT_LANGUAGE	De
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_CONNECTION	Keep-Alive
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)
HTTP_UA_PIXELS	1024x768
HTTP_UA_COLOR	color16
HTTP_UA_OS	Windows NT
GATEWAY_INTERFACE	CGI/1.1
PATH_TRANSLATED	C:\InetPub\wwwroot
REMOTE_ADDR	10.1.1.21
REMOTE_HOST	10.1.1.21
REMOTE_PORT	
SCRIPT_NAME	/scripts/c16_web.dll
SERVER_NAME	10.1.1.22
SERVER_PORT	80
SERVER_PORT_SECURE	0
SERVER_PROTOCOL	HTTP/1.1
SERVER_SOFTWARE	Microsoft-IIS/4.0

Welche von diesen Werten gesetzt sind, kann je nach verwendetem Browser und Installation unterschiedlich ausfallen. Es können weitere Werte existieren oder einzelne Werte nicht gesetzt sein. Wird versucht einen nicht vorhandenen Wert zu lesen, wird ein Leerstring zurückgegeben.

Kontakt

_WseInfoLogPath

Ermittelt den Eintrag von web_log_path

Wert 14

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Eintrag von web_log_path in der Konfigurationsdatei der Web-Schnittstelle.

Kontakt

_WseInfoModulePath

Ermittelt den Eintrag von web_module_path Wert

12

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Eintrag von web_module_path in der Konfigurationsdatei der Web-Schnittstelle.

Kontakt

_WseInfoReqData

Request-Daten

Wert 15

Verwandte

Siehe Befehle,

WseInfo(),

WseInfoReqPath

Option bei WseInfo(). Die Request-Daten werden zurückgegeben.

Wird die Methode GET verwendet, ermittelt die Anweisung

WseInfo(_WseInfoReqData) den vollständigen Request-String.

Bei der Methode POST wird durch die Anweisung der vollständige Request-Body übergeben. Soll der Inhalt einer Datei übergeben werden, muss die Methode POST verwendet werden. Der Inhalt der Datei kann dann über die Anweisung ermittelt werden.

In beiden Fällen darf eine Länge von 4 KB nicht überschritten werden, sonst erfolgt der Laufzeitfehler ErrValueRange. Dies bedeutet auch, dass der Inhalt der Datei nicht größer als 4 KB sein darf.

Beispiel:

HTML-Seite:

```
<form action="C16.URL(absolut)" enctype="multipart/form-data" method="POST"><p><input type="hidde
```

CONZEPT 16-Prozedur:

```
local{ tReqData : alpha(4096) ; }...tReqData # WseInfo(_WseInfoReqData);...
```

Kontakt

_WseInfoReqMethod

Request-Methode

Wert 6

Verwandte

Siehe **Befehle**,

WseInfo()

Option bei **WseInfo()**. Die Request-Methode wird zurückgegeben.

Für die weitere Verarbeitung im Client ist nur die Unterscheidung zwischen einem GET-und einem POST-Request nötig, andere Request-Methoden werden im Client behandelt.

Kontakt

_WseInfoReqPath

Alle Request-Daten

Wert 7

Verwandte

Siehe **Befehle**, **WseInfo()**,

WseInfoReqData

Option bei **WseInfo()**. Der Request-Pfad nach der Schnittstelle wird zurückgegeben.

Wird beim Browser ein Link in der Form

http://www.vectorsoft.de/scripts/c16_web.dll/Factsheet/Server?C16ID=...&Parameter1=...

angeklickt, kann mit diesem Übergabewert die Zeichenkette zwischen der Schnittstelle und dem ersten Parameter ermittelt werden. In unserem Beispiel wird also /Factsheet/Server zurückgegeben.

Kontakt

_WseInfoReqProtocol

Verwendetes Protokoll

Wert 5

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Bei diesem Parameter wird das verwendete Protokoll (HTTP/1.0 oder HTTP/1.1) zurückgegeben.

_WseInfoRootPath

Ermittelt den Eintrag von web_root_path

Wert 11

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt den Eintrag von web_root_path in der Konfigurationsdatei der Web-Schnittstelle.

Kontakt

_WseInfoUserID

ID des Web-Benutzers

Wert 1

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Über diesen Parameter wird die ID des Web-Benutzers ermittelt.

Der Web-Benutzer unterscheidet sich vom Datenbank-Benutzer. Da mehrere Web-Benutzer sich eine Datenbankverbindung teilen können, arbeiten mehrere Web-Benutzer der Applikation mit der gleichen Datenbankbenutzer-ID in der Datenbank. Eine Unterscheidung kann nur über die ID oder die Nummer des Web-Benutzers erfolgen.

Ebenso ist nicht gesichert, dass ein Web-Benutzer für jeden Request die gleiche Verbindung zur Datenbank erhält. Ein Web-Benutzer arbeitet also auch mit unterschiedlichen Datenbank-IDs. Da die Benutzer-ID des Web-Benutzers eine 24stellige Zeichenkette ist, eignet sie sich nicht als Bestandteil von Namen von temporären (sessionbezogenen) Texten oder Selektionen. Zu diesem Zweck sollte die Benutzer-Nummer (_WseInfoUserNumber) verwendet werden.

Kontakt

_WseInfoUserIP

IP-Adresse der Quelle

Wert 2

Verwandte

Siehe **Befehle**,

WseInfo()

Option bei **WseInfo()**. Dies ist die IP-Adresse, die als Quelle des Requests vom Client ermittelt wurde.

Bei Verwendung von Proxy-Servern oder Gateways muss diese Adresse nicht der des Browserrechners entsprechen. Eine Unterscheidung von Anwender-Rechnern anhand der IP-Adresse ist daher nicht immer möglich. Unter Umständen (zum Beispiel bei "reverse hosting") haben alle Requests dieselbe IP-Adresse.

Kontakt

_WseInfoUserNumber

Nummer des Web-Benutzers

Wert 3

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Dieser Parameter ermittelt die eindeutige Nummer des Web-Benutzers innerhalb der Applikation.

Im Unterschied zur Benutzer-ID wird hier eine Zahl zurückgegeben.

Kontakt

_WseInfoUserSessionTime

Zeit seit der Anmeldung

Wert 4

Verwandte

Siehe Befehle,

WseInfo()

Option bei WseInfo(). Es wird die Zeit zurückgegeben, die seit der Anmeldung des Web-Benutzers vergangen ist. Der Befehl liefert dabei die Anzahl der Sekunden zurück. Der Wert wird als Zeichenkette zurückgegeben, muss also zur weiteren Verarbeitung gewandelt werden.

Kontakt

_WseNoParse

Die zurückgegebenen Daten werden nicht durch den Client interpretiert Wert 16 / 0x10

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Der Web-Benutzer kann sofort einen weiteren Request verarbeiten. Diese Option ist automatisch aktiv, wenn der MIME-Typ nicht "text/html" ist.

Kontakt

_WseRetExpires

Setzen des Verfallsdatums für Daten der Web-Schnittstelle Wert 4

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Bei Angabe dieser Option wird das Verfallsdatum für die mit der Web-Schnittstelle übertragenen Daten definiert.

Kontakt

_WseRetFile

Übergabe einer externen Datei

Wert 3

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). In (var4) steht der Pfad und der Name der Datei.

Kontakt

_WseRetHeader

Übertragen des HTTP-Headers mit ergänzten Feldern Wert 0

Verwandte

Siehe [Befehle](#),

WseReturn()

Option bei [WseReturn\(\)](#). Es wird der HTTP-Header übertragen und um zusätzliche Felder ergänzt. Die einzelnen Felder müssen durch Zeilenumbrüche getrennt sein.

Beispiel:

```
'Content-Language: en' . . .
```

Kontakt

_WseRetString

Angabe einer Zeichenkette

Wert 1

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). In (var4) wird eine Zeichenkette angegeben.

Kontakt

_WseRetText

Übertragen eines Textes

Wert 2

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Der Text liegt in einem Textpuffer vor. Der Textpuffer wird in (var4) übergeben.

Kontakt

_WseSendAfter

Die zurückgegebenen Daten werden erst nach dem Beenden der Prozedur an den Web-Server übergeben

Wert 32 / 0x20

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Sollte ein interner Text übergeben werden, ist darauf zu achten, dass der entsprechende Puffer nach Prozedurende noch vorhanden ist.

Diese Art der Rückgabe ist etwas schneller, wenn WseNoParse angegeben wird.

Kontakt

_WseTerm

Die Session wird beendet

Wert 64 / 0x40

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Die Session wird nach der Rückgabe beendet. Der Web-Benutzer wird entfernt und die Benutzer-ID ist anschließend nicht mehr gültig.

Kontakt

_WseUserHTML

Status des Web-Benutzers

Wert 2

Verwandte

Siehe Befehle,

WseStatus()

Option bei WseStatus(). Die Funktion wurde über den Befehl C16.CALL() aus einer HTML-Seite aufgerufen.

Kontakt

_WseUserInit

Status des Web-Benutzers

Wert 0

Verwandte

Siehe Befehle,

WseStatus()

Option bei WseStatus(). Dieser Wert wird zurückgegeben, wenn der Web-Benutzer zum ersten mal die Request-Prozedur aufruft.

Dieser Zustand sollte abgefragt und gegebenenfalls globale Datenbereiche angelegt werden.

Kontakt

_WseUserProc

Status des Web-Benutzers

Wert 1

Verwandte

Siehe Befehle,

WseStatus()

Option bei WseStatus(). Die Request-Prozedur wurde erneut aufgerufen.

Kontakt

_WseUserTerm

Status des Web-Benutzers

Wert 3

Verwandte

Siehe Befehle,

WseStatus()

Option bei WseStatus(). Die Request-Prozedur wird nochmals aufgerufen, wenn der Web-Benutzer vom Client entfernt wird.

Dieser Zustand sollte abgefragt und gegebenenfalls bei _WseUserInit angelegte globale Datenbereiche gelöscht werden.

Druckfunktionen

Funktionen zum Drucken

Befehlsgruppen,

Siehe Befehlsliste,

Beispiel

Befehle

- PrtAdd
- PrtAddByName
- PrtDeviceClose
- PrtDeviceOpen
- PrtDeviceRefresh
- PrtFormClose
- PrtFormOpen
- PrtInfo
- PrtInfoStr
- PrtJobClose
- PrtJobOpen
- PrtJobWrite
- PrtPrinterRefresh
- PrtPropGet
- PrtPropSet
- PrtRtfSearch
- PrtSearch
- PrtUnit
- PrtUnitLog

Konstanten

- PrtAddBoundIgnore
- PrtAddPageBreak
- PrtAddRelative
- PrtAddTop
- PrtCount
- PrtDeviceSystem
- PrtDoc
- PrtDocDinA4
- PrtDocDinA5
- PrtFirst
- PrtFrame
- PrtHeightContent
- PrtHeightFix
- PrtIndex
- PrtInfoBinCount
- PrtInfoBinId
- PrtInfoBinName
- PrtInfoPaperCount
- PrtInfoPaperHeight
- PrtInfoPaperId
- PrtInfoPaperWidth
- PrtJobDoc

- PrtJobOpenRead
- PrtJobOpenTemp
- PrtJobOpenWrite
- PrtJobPage
- PrtJobPageBreak
- PrtJobPageEnd
- PrtJobPageStart
- PrtJobPreview
- PrtJobPreviewPrintClicked
- PrtJobPreviewValidate
- PrtJobPrint
- PrtLast
- PrtNext
- PrtParent
- PrtPrev
- PrtRoot
- PrtRulerLeft
- PrtRulerNone
- PrtRulerTop
- PrtStyleCapFnResult
- PrtStyleCapNormal
- PrtStyleCapPageNo
- PrtType
- PrtTypePrintDoc
- PrtTypePrintDocRecord
- PrtTypePrintForm
- PrtTypePrtRtf
- PrtUnitCentimetres
- PrtUnitInches
- PrtUnitMillimetres
- PrtUnitPoints
- PrtUnitTwips

Kontakt

obj -> PrtPropGet(int1, var2) : logic  Auslesen einer

Eigenschaft eines Druckobjektes

obj Objekt
 Konstante der

int1 Eigenschaft

var2 Wert der Eigenschaft

Resultat logic Funktion erfolgreich

Siehe Verwandte Befehle,
PrtPropSet(),

Alphabetische Liste aller
Eigenschaften

Dieser Befehl liest eine Eigenschaft eines Druckobjektes aus.

Als erster Parameter muss die Konstante der Eigenschaft übergeben werden. Die Konstanten setzen sich aus _PrtProp und dem Namen der Eigenschaft zusammen.

Im zweiten Parameter wird die Variable übergeben, in die der Wert der Eigenschaft kopiert werden soll.

Beispiel:

```
local{ tTitle : alpha;}// Auslesen des Objekttitels$edTitle->PrtPropGet(_PrtPropCaption, tTitle);
```

Das Kommando kann ebenfalls dazu verwendet werden, um zu ermitteln, ob ein bestimmtes Objekt eine Eigenschaft besitzt. Ist eine Eigenschaft nicht vorhanden, liefert der Befehl den Wert false zurück.



Alternativ kann die Eigenschaft auch wie folgt gelesen werden:

Beispiel:

```
local{ tTitle : alpha;}...// Auslesen des ObjektitelstTitle # $edTitle->ppCaption;
```

Kontakt

obj -> PrtPropSet(int1, var2[, int3]) : logic  Auslesen einer

Eigenschaft eines Druckobjektes

obj Objekt

 Konstante der

int1 Eigenschaft

var2 Zu setzender Wert

int3 Zusätzliche Information

Resultat logic Funktion erfolgreich

Verwandte Befehle,
PrtPropGet(),

Siehe

Alphabetische Liste aller
Eigenschaften

Dieser Befehl setzt eine Eigenschaft eines Druckobjektes.

Als erster Parameter muss die Konstante der Eigenschaft übergeben werden. Die Konstanten setzen sich aus _PrtProp und dem Name der Eigenschaft zusammen.

Im zweiten Parameter wird der zu setzende Wert übergeben.

Beispiel:

```
// Setzen des Objekttitels auf "Neu"$Text->PrtPropSet(_PrtPropCaption, 'Neu');
```



Alternativ kann die Eigenschaft auch wie folgt gesetzt werden:

Beispiel:

```
// Setzen des Objekttitels auf "New"$Text->ppCaption # 'New';
```

Ein Vorteil neben der kompakteren Schreibweise gegenüber PrtPropSet und PrtPropGet() besteht darin, dass schon während der Kompilierung eine Typprüfung vorgenommen wird:

```
$Object->PrtPropSet(_PrtPropCaption, 100);
```

liefert während der Laufzeit den Rückgabewert false, da 100 vom Typ int ist.

```
$Object->ppCaption # 100;
```

erzeugt bereits während der Kompilierung der entsprechenden Prozedur einen Fehler. Typ-Fehler dieser Art werden somit vermieden. Beim Lesen einer Eigenschaft, die im referenzierten Objekt nicht vorhanden ist, wird ein Laufzeitfehler erzeugt. Der Laufzeitfehler kann durch die Kapselung in einem try-Block unterbunden werden.

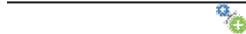
Beispiel

```
try{ ErrTryCatch(_ErrPropInvalid, true); $Objekt->ppCaptionInt # 100; ... }
```

Der optionale Parameter (int3) muss nur angegeben werden, wenn zusätzliche Informationen einer Eigenschaft zugeordnet werden können. Entsprechende Hinweise befinden sich in den Beschreibungen der Eigenschaften.

Kontakt

PrtJobOpen(alpha1[, alpha2[,
int3[, int4[, alpha5[, int6]]]]])
: handle Druckjob
anlegen



	Druck-Dokument Name oder	
alpha1	<u>PrtDocDinA4</u> Din A4 Seitenformat	
	<u>PrtDocDinA5</u> Din A5 Seitenformat	
alpha2	Externer Dateiname	
	Optionen	Druckjob zum
	<u>PrtJobOpenWrite</u>	Schreiben öffnen
	<u>PrtJobOpenRead</u>	Druckjob zum
	<u>PrtJobOpenTemp</u>	Lesen öffnen Temporären
int3	<u>PrtJobOpenSort</u>	Druckjob öffnen Visuelle Reihenfolge der Druckobjekte
	<u>PrtJobOpenEmbedImages</u>	Bilder einbetten
	<u>PrtJobOpenVerbose</u>	Druckjob mit zusätzlichen Informationen anlegen

	Objekttyp	
	<u>PrtTypePrintDoc</u>	PrintDoc-Objekt
int4		drucken

	<u>PrtTypePrintDocRecord</u>	PrintDocRecord-Objekt
		drucken

alpha5	Krypt-Key	
	Unicode-Unterstützung	

	<u>PrtConvNone</u>	Das Objekt laden, wie es
		gespeichert wurde.

int6	<u>PrtConvUnicode</u>	Unicode-Unterstützung
		aktiviert

	<u>PrtConvAnsi</u>	keine
		Unicode-Unterstützung

Resultat handle	Objekt	
	<u>Verwandte Befehle, PrtJobWrite()</u>	

Siehe
[PrtJobClose\(\)](#), [PrintJob](#), [Beispiel](#)

Mit diesem Befehl wird ein neuer Druckjob angelegt oder aus einer externen Datei gelesen. Als Resultat wird ein Deskriptor auf ein [PrintJob](#)-Objekt zurückgegeben. Beim Schließen des Druckjobs mit dem Befehl [PrtJobClose\(\)](#) wird bestimmt, ob eine Druckvorschau angezeigt werden soll.

Der Name des zu druckenden Dokuments wird in (alpha1) übergeben. Wird das Dokument aus mehreren [PrintForm](#)-Objekten zusammengestellt, werden in (alpha1) entweder die Konstanten [PrtDocDinA4](#), [PrtDocDinA5](#) oder ein [PrintDoc](#)-Objekt mit

Kontakt

einem Seitenformat angegeben.

Wird als Option (int3) PrtJobOpenWrite angegeben, wird der Druckjob zum Schreiben geöffnet. Zusätzlich zu den Druckjobdaten wird die Versionsnummer der verwendeten c16_objw.dll gespeichert. Mit der Option PrtJobOpenRead kann ein gespeicherter Druckjob gelesen werden. Die Version des Druckjobs kann dann über die Eigenschaft Version ermittelt werden. Diese ist 0, wenn der Druckjob vor der 5.8.04 erzeugt wurde.

Im Parameter (int4) wird angegeben, ob in (alpha1) ein PrintDoc-Objekt oder ein PrintDocRecord-Objekt übergeben wurde. Werden PrintForm-Objekte verwendet muss PrtTypePrintDoc angegeben werden.

Im Argument (alpha5) kann ein bis zu 64 Zeichen langes Passwort angegeben werden, welches zum Verschlüsseln des Druckjobs verwendet wird.

Im Parameter (int6) wird angegeben, ob ein Druckjob mit Unicode-Unterstützung erzeugt werden soll. Der Parameter ist nur beim Schreiben eines Druckjobs (PrtJobOpenWrite) zulässig. Es können folgende Konstanten übergeben werden:

- PrtConvNone

Es wird ein Druckjob mit der in der Datenbank gespeicherten Unicode-Unterstützung des in (alpha1) angegebenen Objekts erzeugt. Es findet keine Konvertierung des Objekts statt. Die Objekte PrtDocDinA4 und PrtDocDinA5 haben keine Unicode-Unterstützung. Soll mit diesen Objekten ein Druckjob mit Unicode-Unterstützung erzeugt werden, muss die Konstante PrtConvUnicode angegeben werden.

- PrtConvUnicode

Es wird ein Druckjob mit Unicode-Unterstützung erzeugt. Ist das angegebene Objekt ohne Unicode-Unterstützung in der Datenbank gespeichert, findet eine Konvertierung des Objekts statt. Dem Druckjob können nur noch Objekte mit Unicode-Unterstützung hinzugefügt werden.

- PrtConvAnsi

Es wird ein Druckjob ohne Unicode-Unterstützung erzeugt. Ist das angegebene Objekt mit Unicode-Unterstützung in der Datenbank gespeichert, findet eine Konvertierung des Objekts statt. Dem Druckjob können nur noch Objekte ohne Unicode-Unterstützung hinzugefügt werden.

Wird keine der Konstanten angegeben, findet keine Konvertierung statt (PrtConvNone).

Der Befehl liefert einen Deskriptor auf den erzeugten oder geöffneten Druckjob zurück, sofern der Aufruf erfolgreich war. Ist ein Fehler aufgetreten wird 0 zurückgegeben.

- Erstellen eines temporären Druckjobs

Kontakt

Der Druckjob wird in eine temporäre externe Datei geschrieben. Die Datei wird nach dem Drucken automatisch wieder entfernt.

Es wird eine temporäre Datei angelegt, wenn in (alpha2) kein externer Dateiname oder in (int3) die Option PrtJobOpenTemp angegeben wird. Werden beide Angaben gemacht, wird der externe Dateiname in (alpha2) ignoriert. Die temporäre Datei wird im Windows-Temp-Verzeichnis angelegt und beim Beenden der Druckvorschau bzw. nach dem Ausdruck wieder gelöscht. Der Name der temporären Datei setzt sich aus C16_-, einer beliebigen achtstelligen hexadezimalen Zahl und der Erweiterung .tmp zusammen. Mit der Option PrtJobOpenWrite wird der Druckjob zum Schreiben geöffnet.

Beispiel:

```
// Temporären Druckjob öffnetPrtJob # PrtJobOpen('Brief', '', _PrtJobOpenWrite | _PrtJobO
```

- **Schreiben einer Druckjob-Datei**

Soll der Druckjob über einen längeren Zeitraum erhalten bleiben, zum Beispiel zur Archivierung, kann er in eine externe Datei geschrieben werden. Dazu muss in (alpha2) der Name und der Pfad der externen Datei angegeben werden.

Mit der Option PrtJobOpenWrite wird der Druckjob zum Schreiben geöffnet. Beispiel:

```
// Druckjob öffnetPrtJob # PrtJobOpen('Brief', 'C:\c16\Brief.job', _PrtJobOpenWrite, _Prt
```

Die Druckjob-Datei kann anschließend weiter verarbeitet werden. Zum Beispiel als binäres Objekt in die Datenbank importiert oder über das PrtJobPreview-Objekt in der Applikation angezeigt werden.

Soll die Datei verschlüsselt gespeichert werden, kann in (alpha5) ein bis zu 64 Zeichen langer Schlüssel angegeben werden.

- **Öffnen einer Druckjob-Datei**

Zum Lesen eines Druckjobs muss in (alpha2) der vollständige Pfad- und Dateiname der externen Druckjob-Datei angegeben werden. Im Argument (int3) muss PrtJobOpenRead übergeben werden.

```
// Druckjob öffnetPrtJob # PrtJobOpen('Brief', 'C:\c16\Brief.job', _PrtJobOpenRead, _PrtT
```

Handelt es sich um einen verschlüsselten Druckjob muss ferner das korrekte Passwort im Argument (alpha5) angegeben werden. Ist das Passwort falsch oder der Druckjob unverschlüsselt und ein Passwort angegeben, wird 0 zurückgegeben.

- **Schreiben einer PDF-Datei**

Aus einem Druckjob kann direkt ein PDF-Dokument erstellt werden (siehe PrtJobPdf). Standardmäßig erfolgt die Ausgabe der Druckobjekte in umgekehrter Reihenfolge ihrer Erstellung. Die Reihenfolge der Druckobjekte spielt beim Drucken keine Rolle. Wird allerdings ein PDF-Dokument erstellt,

Kontakt

muss die Reihenfolge der Druckobjekte der Darstellung auf der Seite folgen, damit innerhalb des PDF-Dokuments Text markiert werden kann. Durch die Angabe der Option

PrtJobOpenSort in (int3) wird die visuelle Reihenfolge definiert. Die Objekte werden dann abhängig von ihrer Position auf der Seite (von oben nach unten und von links nach rechts) in das PDF-Dokument geschrieben. Soll unabhängig von ihrer Position bestimmte Objekte über andere Objekte gedruckt werden, muss die Eigenschaft ZOrder verwendet werden.

In dem Druckjob können erweiterte Informationen abgelegt werden, wenn zusätzlich die Konstante PrtJobOpenVerbose in (int3) angegeben wird. Die zusätzlichen Informationen werden bislang nicht ausgewertet.

Sollen die Bilder und Dokumente, die innerhalb eines Druckjobs angezeigt werden auch innerhalb des Druckjobs gespeichert werden, muss im Parameter (int3) die Konstante PrtJobOpenEmbedImages angegeben werden.



Als Resultat wird der Deskriptor auf den geöffneten / angelegten Druckjob zurückgegeben. Konnte der Druckjob nicht geöffnet werden, wird als Resultat 0 zurückgegeben. In diesem Fall wird der globale Fehlercode gesetzt, der mit ErrGet() ermittelt werden kann. Beispielsweise könnte das Öffnen aus folgenden Gründen fehlschlagen:

- Das Druck-Dokument (alpha1) ist nicht vorhanden.
- Der Pfad oder die Datei (alpha2) ist nicht vorhanden oder kann nicht erstellt werden.

Kontakt

obj ->

PrtJobWrite(handle1) : 

int

Druckjob schreiben

obj Printjob-Objekt

Seiten-Deskriptor oder

PrtJobDoc

Dokument

schreiben

PrtJobPageStart Erste Seite

erzeugen

handle1 PrtJobPageBreak Seite

abschließen

und neue

Seite

erzeugen

PrtJobPageEnd

Letzte Seite

abschließen

Resultat int Fehlernummer oder

Seitendeskriptor

Verwandte Befehle,

Siehe

PrtJobOpen(), Beispiel

Mit dem Befehl wird in einen durch PrtJobOpen() erzeugten Druckjob geschrieben.

Das erste Argument enthält den Deskriptor des Druckjobs.

Wird im Argument (handle1) der Deskriptor eines Seiten-Objektes angegeben, wird diese Seite in den Druckjob geschrieben. Der Druckjob kann so seitenweise zusammengestellt werden. Folgende Konstanten können angegeben werden:

- PrtJobDoc

Das gesamte Dokument wird in den Druckjob geschrieben.

- PrtJobPageStart

Es wird eine neue Seite erzeugt.

- PrtJobPageBreak

Die aktuelle Seite wird in den Druckjob geschrieben und eine neue Seite erzeugt.

- PrtJobPageEnd

Die aktuelle Seite wird in den Druckjob geschrieben.



Auf eine geschriebene Seite kann nicht mehr zugegriffen werden. Änderungen sind dann nicht mehr möglich.

Der Rückgabewert definiert eine Fehlernummer. Wurde eine der Konstanten

PrtJobPageStart oder PrtJobPageBreak übergeben, wird der Deskriptor auf die neu erzeugte Seite zurückgegeben.

Beispiel:

Kontakt

```
// Bereits verbrauchter Platz auf SeiteAddSize # tPage->ppBoundAdd;// Maximal zu Verfügung stehe
```

Kontakt

obj -> PrtJobClose(int1[,
handle2]) : int Druckjob
schließen



obj Printjob-Objekt

Optionen (optional)

PrtJobPreview

Vorschau anzeigen

PrtJobPreviewValidate Vorschau anzeigen

und prüfen, ob der
Druck-Dialog mit OK

verlassen wurde

sofort drucken

PrtJobPrint als PDF-Datei

PrtJobPdf drucken

PrtJobTif als

PrtJobTif MultiPage-Tiff-Datei

int1

drucken

PrtJobEmf als EMF-Datei(en)

PrtJobXml drucken

PrtJobHidden als XML-Datei

PrtJobCancel drucken

Fortschrittsanzeige

während der

Druckaufbereitung

unterdrücken

PrtJobCancel Druckaufbereitung

abbrechen

handle2 PrtDevice-Objekt (optional)

Resultat int Fehlerwert

Verwandte Befehle, PrtJobOpen(),

Siehe PrtDeviceOpen(), Beispiel

Mit dem Befehl wird ein durch PrtJobOpen() geöffneter oder erzeugter Druckjob geschlossen.

Als (obj) wird der von PrtJobOpen() zurückgegebene Deskriptor übergeben.

Wird im Argument (int1) die Konstante PrtJobPreview angegeben, wird die Druckvorschau gestartet. Die Konstante PrtJobPreviewValidate bewirkt, dass die Druckvorschau gestartet wird und geprüft wird, ob der Druck-Dialog mit OK verlassen wurde. Ist dies der Fall und beim Drucken ist kein Fehler aufgetreten, wird der Wert PrtJobPreviewPrintClicked zurückgegeben.

Mit PrtJobPrint wird der Druckjob auf einen Drucker geleitet. Hierzu kann im Argument (handle2) der Deskriptor eines PrintDevice-Objekts übergeben werden, der vom Befehl PrtDeviceOpen() zurückgegeben wurde. Wird in (handle2) kein PrintDevice übergeben, erfolgt die Ausgabe auf den Standard-Drucker.

Das Dokument muss nicht auf einen Drucker ausgegeben werden. Mit den folgenden Parametern, die als (int1) übergeben werden, können auch andere Formate erzeugt werden:

Kontakt

- PrtJobPdf

Es wird ein PDF-Dokument erzeugt. Beim PrintJob-Objekt muss mindestens die Eigenschaft PdfFileName angegeben werden.

- PrtJobTif

Es wird ein Multipage-TIFF-Dokument erzeugt. Beim PrintJob-Objekt muss die Eigenschaft TifFileName angegeben werden.

- PrtJobEmf

Es wird für jede Seite ein EMF-Dokument erzeugt. Beim PrintJob-Objekt muss die Eigenschaft EmfFileName angegeben werden. Besteht der Druckjob aus mehreren Seiten, wird dem Dateiname jeweils die Seitennummer angehängt.

- PrtJobXml

Es wird ein XML-Dokument erzeugt. Beim PrintJob-Objekt muss mindestens die Eigenschaft XmlFileName angegeben werden. Der Zeichensatz der XML-Datei wird über die Eigenschaft Charset angegeben.

Die folgenden Konstanten können kombiniert werden, um mehrere Formate gleichzeitig zu generieren:

- PrtJobPreview bzw. PrtJobPreviewValidate
- PrtJobPrint
- PrtJobPdf
- PrtJobEmf
- PrtJobTif
- PrtJobXml

Die Anzeige der Druckvorschau bzw. die Formatgenerierung erfolgt entsprechend der oben angegebenen Reihenfolge. Falls bei der Generierung eines Formates ein Fehler auftritt, wird nicht abgebrochen, sondern mit der Erstellung des nächsten Formates fortgefahrene. Der Rückgabewert entspricht dem Fehlerwert, des zuerst aufgetretenen Fehlers.

Zusätzlich können die PrtJob...-Konstanten mit PrtJobHidden kombiniert werden, um die Fortschrittsanzeige beim Druckvorgang zu unterdrücken.

Über die Eigenschaft PrintToFile des PrintJob-Objekts kann die Druckausgabe auf Datei umgeleitet werden.

Das Argument (int1) kann entfallen. Insbesondere bei temporären Druckjobs sollte jedoch eine der Konstanten angegeben sein.

Wird in (int1) die Konstante PrtJobCancel angegeben, wird der Druckjob geschlossen, ohne das es zu einem Ausdruck oder der Anzeige einer Druckvorschau kommt. Die temporär angelegten Dateien werden gelöscht.

Beispiel:

```
// Ausgabeeinheit öffnetDev # PrtDeviceOpen('HP LaserJet', _PrtDeviceSystem); // Temporären Druck
```

Der Befehl liefert folgende Resultate zurück:

[ErrOk](#)

Die Verarbeitung des Druckjobs war erfolgreich.

[ErrRange](#)

Keine zu druckende Seite oder Startseite größer Endeseite.

[ErrSystem](#)

Beim Drucken ist ein Fehler aufgetreten, der nicht weiter spezifiziert werden kann.

[PrtJobPreviewPrintClicked](#)

Die Option PrtJobPreviewValidate wurde angegeben und im Druck-Dialog wurde auf OK geklickt.

[ErrPrtPaperFormat](#)

Das Papierformat des PrintDoc-Objektes konnte auf der angegebenen PrintDevice nicht ausgegeben werden.

[ErrPdfPageAppend](#)

Es konnte keine Seite an das PDF-Dokument angehängt werden.

[ErrPdfInsertMetaFile](#)

Der Seite des PDF-Dokuments konnte kein Inhalt hinzugefügt werden.

[ErrPdfNotPdfA](#)

Es sollte ein PDF/A-Dokument erstellt werden. Dieser Fehler wird zurückgegeben, wenn nicht alle Informationen für ein PDF/A-Dokument geschrieben werden konnten.

[ErrFsi...](#)

Beim Erstellen der externen Datei ist ein Problem aufgetreten. Die entsprechenden Rückgabewerte sind im Abschnitt Externe Dateioperationen erläutert.

Kontakt

PrtFormOpen(int1[, alpha2[,
int3]]) : handle Print-Objekt
öffnen



	Objekttyp	
int1	<u>PrtTypePrintForm</u>	PrintForm-Objekt
	<u>PrtTypePrintFormList</u>	PrintFormList-Objekt
	<u>PrtTypePrintDoc</u>	PrintDoc-Objekt alpha2

Objektname

	Optionen (optional)	
int3	<u>PrtConvNone</u>	Das Objekt laden, wie es gespeichert wurde.
	<u>PrtConvUnicode</u>	Unicode-Unterstützung aktiviert
	<u>PrtConvAnsi</u>	keine Unicode-Unterstützung

Resultat handle Deskriptor

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

Siehe [PrtAddByName\(\)](#), [PrtFormClose\(\)](#),
[Beispiel](#)

Der Befehl öffnet ein [PrintForm](#)-, [PrintFormList](#)- oder [PrintDoc](#)-Objekt, welches in der Datenbank abgelegt ist. In (int1) wird der Typ des zu ladenden Objektes übergeben. Der Objektnname (alpha2) ist der Name, unter dem das Formular oder Dokument in der Datenbank abgelegt ist.

Optional kann in (int3) angegeben werden, ob das Objekt mit Unicode-Unterstützung geladen werden soll. Folgende Konstanten können übergeben werden:

- [PrtConvNone](#)

Das angegebene Objekt wird mit der in der Datenbank gespeicherten Unicode-Unterstützung geladen. Es findet keine Konvertierung des Objekts statt.

- [PrtConvUnicode](#)

Das angegebene Objekt wird mit Unicode-Unterstützung geladen. Ist das Objekt ohne Unicode-Unterstützung in der Datenbank gespeichert, findet eine Konvertierung des Objekts statt.

- [PrtConvAnsi](#)

Das angegebene Objekt wird ohne Unicode-Unterstützung geladen. Ist das Objekt mit Unicode-Unterstützung in der Datenbank gespeichert, findet eine Konvertierung des Objekts statt.

Wird keine der Konstanten angegeben, werden die Objekte ohne eine Konvertierung geladen ([PrtConvNone](#)).

Nachdem das Formular oder Dokument geöffnet wurde, können seine Eigenschaften verändert und mit [PrtAdd\(\)](#) einem Druckjob hinzugefügt werden. Stimmt die Unicode-Unterstützung des geladenen Objekts nicht mit der Unicode-Unterstützung

Kontakt

des Druckjobs überein, wird der Laufzeitfehler [ErrHdlInvalid](#) erzeugt.



Der Suchpfad schließt nicht das neu geladene Druck-Objekt ein. Ein Zugriff auf die Elemente des Formulars bzw. des Dokumentes ist erst nach dem Setzen des Suchpfades mit [WinSearchPath\(\)](#) oder mit dem Befehl [PrtSearch\(\)](#) möglich. In beiden Fällen kann der von [PrtFormOpen](#) zurückgegebene Deskriptor als Startpunkt der Suche verwendet werden. Es kann auch die Anweisung [with](#) verwendet werden, um über die Namen der Objekte auf diese Objekte innerhalb des Druck-Objekts zuzugreifen.

Mit dem Befehl [PrtFormClose\(\)](#) wird das Objekt wieder geschlossen.

Beispiel:

```
// Temporären Druckjob öffnen (PrintDocument DinA4, leer) tJob # PrtJobOpen(_PrtDocDinA4, '', _Prt
```

Kontakt

obj -> PrtFormClose() : logic  PrintForm-

/PrintDoc-Objekt schließen

obj Deskriptor

Resultat logic Erfolgsstatus

Verwandte Befehle,

Siehe

PrtFormOpen(),

Beispiel

Mit diesem Befehl wird ein mit PrtFormOpen() geöffnetes PrintForm-Objekt wieder geschlossen.

Der von PrtFormOpen() zurückgegebene Deskriptor wird in (obj) übergeben.

Beispiel:

```
// Temporären Druckjob öffnen (PrintDocument DinA4, leer)tJob # PrtJobOpen(_PrtDocDinA4, '', _Prt
```

Kontakt

obj -> PrtAdd(handle1[, int2[, int3[,
int4]]]) : int



PrintForm zu einem Printjob hinzufügen

obj Seite des PrintJob-Objektes

handle1 PrintForm-Objekt

Optionen

PrtAddPageBreak automatischer

Seitenumbruch

anderes

Druckobjekt

überdrucken

PrtAddRelative Relative Angabe

der

Druckposition

int3 Abstand vom Seitenanfang

int4 Abstand vom linken Seitenrand

Resultat int Seitenumbruch-Flag

Verwandte Befehle,

Siehe PrtAddByName(), Beispiel, Drucken mit Wasserzeichen (Blog)

Der Befehl fügt die in einem PrintForm-Objekt (handle1) enthaltenen Druckobjekte zu einer Seite (obj) des Druckjobs an der aktuellen Druckposition hinzu. Wird im Argument (int2) die Konstante PrtAddPageBreak übergeben, wird anhand der aktuellen Druckposition und des umschließenden Rechtecks der hinzuzufügenden Objekte ermittelt, ob automatisch ein Seitenumbruch erfolgt oder nicht.

Der Seitenumbruch erfolgt immer vor dem Hinzufügen der PrintForm. Soll zum Beispiel vor dem Seitenumbruch eine Seitennummer geschrieben werden, reicht es nicht aus den Rückgabewert von PrtAdd auszuwerten. Die Vorgehensweise in diesem Fall ist in einem Beispiel beschrieben.

Soll ein bereits an dieser Stelle gedrucktes Objekt überdruckt werden, kann die Konstante PrtAddTop angegeben werden.

Die Position des PrintForm-Objekts auf der Seite kann über die Parameter (int3) und (int4) festgelegt werden. Die Angabe der Position erfolgt in logischen Einheiten. Die Einheiten können mit dem Befehl PrtUnitLog() von einer beliebigen Einheit in logische Einheiten umgerechnet werden.

Soll das Objekt relativ zur aktuellen Druckposition positioniert werden, muss in (int2) PrtAddRelative angegeben werden.

Der Rückgabewert ist 1, wenn ein Seitenumbruch erfolgte und die Konstante PrtAddPageBreak angegeben wurde, sonst 0.



Ist bei untergeordneten Objekten die Eigenschaft FontParent auf true gesetzt, wird der Font vom PrintForm-Objekt nur verwendet, wenn auch bei diesem die Eigenschaft FontParent auf true gesetzt ist. Sonst wird der Font von der Job-Seite (siehe PrtJobWrite()) verwendet.

Kontakt

Beispiel:

```
// PrintForm an die Seite anhängenPage->PrtAdd(tPrintForm); // PrintForm an diese oder die nächst
```

Kontakt

obj -> PrtAddByName(alpha1[, int2[, int3,
int4[, int5[, int6]]]]) : int PrintForm zu einem
Printjob hinzufügen



obj	Seite des PrintJob-Objektes
alpha1	PrintForm-/PrintFormList-Name Optionen (optional)
	<u>PrtAddPageBreak</u> automatischer Seitenumbruch
int2	<u>PrtAddTop</u> anderes Druckobjekt <u>PrtAddRelative</u> überdrucken relative Angabe der Druckposition
int3	Abstand vom Seitenanfang (optional)
int4	Abstand vom linken Seitenrand (optional) Unicode-Unterstützung (optional)
	<u>PrtConvNone</u> Das Objekt laden, wie es gespeichert wurde.
int5	<u>PrtConvUnicode</u> Unicode-Unterstützung <u>PrtConvAnsi</u> aktiviert keine Unicode-Unterstützung
int6	Objekttyp (optional) <u>PrtTypePrintForm</u> PrintForm -Objekt <u>PrtTypePrintFormList</u> PrintFormList-Objekt

Resultat int Seitenumbruch-Flag

Siehe Verwandte Befehle, PrtAdd()

Die Funktionsweise des Befehls ist identisch zu PrtAdd(), jedoch wird anstelle des PrintForm-Deskriptors der Name einer PrintForm in (alpha1) angegeben.

Die Position des PrintForm-Objekts auf der Seite kann über die Parameter (int3) und (int4) festgelegt werden. Die Angabe der Position erfolgt in logischen Einheiten. Die Einheiten können mit dem Befehl PrtUnitLog() von einer beliebigen Einheit in logische Einheiten umgerechnet werden.

Soll das Objekt relativ zur aktuellen Druckposition positioniert werden, muss in (int2) PrtAddRelative angegeben werden.

Optional kann mit dem Parameter (int5) das Objekt mit oder ohne Unicode-Unterstützung geladen werden. Folgende Konstanten können übergeben werden:

- PrtConvNone

Das angegebene Objekt wird mit der in der Datenbank gespeicherten Unicode-Unterstützung geladen. Es findet keine Konvertierung des Objekts statt.

- PrtConvUnicode

Kontakt

Das angegebene Objekt wird mit Unicode-Unterstützung geladen. Ist das Objekt ohne Unicode-Unterstützung in der Datenbank gespeichert, findet eine Konvertierung des Objekts statt.

- **PrtConvAnsi**

Das angegebene Objekt wird ohne Unicode-Unterstützung geladen. Ist das Objekt mit Unicode-Unterstützung in der Datenbank gespeichert, findet eine Konvertierung des Objekts statt.

Wird keine der Konstanten angegeben, werden die Objekte ohne eine Konvertierung geladen (PrtConvNone).

Das Objekt muss mit der gleichen Unicode-Einstellung geladen werden, wie der Druckjob zu dem das Objekt hinzugefügt werden soll, angelegt wurde. Wird einem Druckjob mit Unicode-Unterstützung ein Objekt ohne Unicode-Unterstützung hinzugefügt (oder umgekehrt), kommt es zu dem Laufzeitfehler ErrHdlInvalid.

Im Argument (int6) muss der zum Objekt (alpha1) passende Objekttyp angegeben werden. Ist das Argument nicht angegeben, wird PrtTypePrintForm verwendet.

Beispiel:

```
tJob # PrtJobOpen(_PrtDocDinA4, '', _PrtJobOpenWrite | _PrtJobOpenTemp, _PrtTypePrintDoc, '', _Pr
```

Kontakt

obj -> PrtInfo(int1[, int2[,



int3]]) : int

Objektinformationen ermitteln

obj Objekt

int1 Optionen (siehe Text)

int2 Count (Optional)

int3 Typ (Optional)

Resultat int Gewünschte Information

Verwandte Befehle,

Siehe

[PrtInfoStr\(\)](#), [WinInfo\(\)](#), [Blog](#)

Der Befehl ermittelt Informationen zu dem angegebenen Referenz-Objekt (obj). Als Referenz können Druck-Objekte übergeben werden.

Je nach übergebenen Parameter in (int1) werden unterschiedliche Resultate

zurückgegeben:

- PrtType Rückgabe des Objekttyps
- PrtRoot Rückgabe des Wurzelobjektes
- PrtParent Rückgabe des Elternobjektes
- PrtFirst Rückgabe des ersten untergeordneten Objektes
- PrtLast Rückgabe des letzten untergeordneten Objektes
- PrtPrev Rückgabe des vorhergehenden Objektes
- PrtNext Rückgabe des nächsten Objektes
- PrtCount Rückgabe der Anzahl der untergeordneten Objekte
- PrtFrame Rückgabe des Druckvorschau-Dialog-Objektes
- PrtIndex Rückgabe des Objekt-Index
- PrtDoc Rückgabe des Druckdokumentes
- PrtJobPage Rückgabe des aktuellen Seiten-Objektes
- PrtJobPageCount Rückgabe der Anzahl der Seiten in einem Print-Job
- PrtInfoPaperCount Anzahl der vorhandenen Papierformate
- PrtInfoPaperID Nummer des Papierformates
- PrtInfoPaperWidth Breite des Papierformates
- PrtInfoPaperHeight Höhe des Papierformates
- PrtInfoBinCount Rückgabe der Anzahl der Druckerschächte
- PrtInfoBinID Rückgabe der Nummer eines Druckerschachtes
- PrtInfoDpiCount Rückgabe der Anzahl der Druckauflösungen
- PrtInfoDpiX Horizontale Druckauflösung
- PrtInfoDpiY Vertikale Druckauflösung

Beispiele:

```
// Ermitteln aller auf dem System installierten DruckertPrinterList # _App->ppPrinterList(_PrtList  
// Alle Druckauflösungen des Standarddruckers ermittelntDevice # PrtDeviceOpen();if (tDevice > 0)
```

Kontakt

obj -> **PrtInfoStr(int1[, int2])** :



alpha

Ermitteln von Informationen

obj Objekt

9
Mode

PrtInfoBinName

Name des

int1

Name des Druckerschachts

PrtInfoPaperName

Name des

11

Panierformats

int2 Index

4 paper formats

Resultat alpha Gewijnschte Information

Siehe Verwandte Befehle, PrtInfo()

Der Befehl ermittelt Informationen über ein gegebenes Objekt. Im Argument (int1) können folgende Konstanten übergeben werden:

PrtInfoBinName Rückgabe des Drucker-Schachtnamens

PrtInfoPaperName Rückgabe des Papierformat-Namens

Kontakt

obj -> PrtSearch(alpha1) : handle | | | | | | Suchen eines Druck-

Objekts über den Namen

obj Startobjekt der Suche
Name des zu suchenden

alpha1 Objekts

Resultat handle Deskriptor des Objekts

Siehe Verwandte Befehle, Blog

PrtSearch() liefert den Deskriptor des gesuchten Objekts. In (alpha1) wird der Name des gesuchten Objekts übergeben. Im Namen können die Wildcard-Operatoren '*' und '?' angegeben werden und werden bei der Suche entsprechend berücksichtigt.

Als Startobjekt wird das Objekt (Druck-Job, DruckerListe ...) angegeben bei dem die Suche gestartet werden soll. Es werden das angegebene Objekt und alle untergeordneten Objekte nach dem übergebenen Namen durchsucht.

Je nach Objekt, das gesucht wird, muss ein entsprechendes Start-Objekt übergeben werden. Wird nach einem bestimmten Windows-Drucker gesucht, muss die Druckerliste angegeben werden. Befindet sich das gesuchte Objekt in einem PrintDoc-oder PrintDocRecord-Objekt, kann der Deskriptor des Printjobs angegeben werden. Ist das Objekt in einem PrintForm-Objekt enthalten, muss der Deskriptor auf das PrintForm-Objekt übergeben werden. Ebenso können Objekte als Startobjekte übergeben werden, die den genannten Objekten untergeordnet sind.

Als Resultat wird der Deskriptor des gefundenen Objekts zurückgegeben. Wurde kein Objekt gefunden ist das Resultat 0.

Beispiele:

```
// Suchen eines Druckers in einer DruckerlistetPrinterList # _App->ppPrinterList;tPrinter # tPrin
```

Kontakt

obj -> PrtRtfSearch(alpha1[, int2[, range3[, alpha4[, var alpha5]]]]) : int



Zeichenfolge in einem Text suchen / Suchen und Ersetzen
obj Deskriptor des PrtRtf-Objekts

alpha1 Suchtext

Optionen (optional)
PrtRtfSearchUp

Suchbereich vom
Ende nach Vorne
durchsuchen
Groß-/Kleinschreibung

int2 PrtRtfSearchWord

beachten
Nur ganze Wörter

PrtRtfSearchReplace Suchbegriff durch

(alpha4) ersetzen
Bereich in (range3)
entfernen

range3 Suchbereich (optional)

alpha4 Ersetzungstext (optional)

Auf Suchtext folgende Zeichenkette

var alpha5 (Optional)

Resultat int Position der gefundenen Zeichenfolge

Siehe Verwandte Befehle, WinRtfSearch()

Diese Funktion durchsucht einen Text in einem PrtRtf-Objekt. Der Deskriptor des Objektes

wird in (obj) angegeben.

Die zu suchende Zeichenkette wird in (alpha1) angegeben. Alle weiteren Parameter sind optional.

Werden keine weiteren Parameter angegeben, wird der gesamte Text von vorne nach hinten nach der Zeichenkette durchsucht. Wird der Begriff gefunden, wird die Position des ersten Zeichens innerhalb des Textes zurückgegeben. Befindet sich die zu suchende Zeichenkette mehrfach im Text, wird nur das erste Vorkommen zurückgegeben. Ist die Zeichenkette nicht vorhanden wird der Wert -1 zurückgegeben.

Beispiel:

```
tPos # tPrtRtf->PrtRtfSearch('suche');
```

Die Suche kann mit folgenden Optionen beeinflusst werden:

- PrtRtfSearchUp

Der Suchbereich wird vom Ende zum Anfang durchsucht.

- PrtRtfSearchCase

Die Groß- und Kleinschreibung des Suchbegriffes wird beachtet.

- PrtRtfSearchWord

Es wird nur nach ganzen Wörtern gesucht.

Kontakt

- PrtRtfSearchReplace

Der Suchtext wird durch den Ersetzungstext in (alpha4) ersetzt.

- PrtRtfSearchDelete

Der Bereich in (range3) wird aus dem RTF-Text entfernt.

Die Optionen können miteinander kombiniert werden. Der zu durchsuchende Text kann durch die Angabe eines Bereiches in (range3) bestimmt werden. Standardmäßig wird der Bereich (0, -1) (Anfang bis Ende) durchsucht.

Wird die Option PrtRtfSearchReplace angegeben, muss in (alpha4) ein entsprechender Ersetzungstext angegeben werden, da sonst der Suchbegriff aus dem Text entfernt wird. Wie beim Suchen wird nur die erste Fundstelle ersetzt.

Wird die Option PrtRtfSearchDelete angegeben, kann zusätzlich mit der Option PrtRtfSearchReplace der Bereich durch einen anderen Text ersetzt werden. Der entsprechende Ersetzungstext wird in (alpha4) angegeben.

Es ist zu beachten, dass durch die Textersetzung eine Verlängerung oder Verkürzung des Textes entsteht. Unter Umständen muss deshalb der Suchbereich (range3) neu gesetzt werden, da diese bestimmt welcher Text-Ausschnitt angezeigt wird.

Wird der optionale var-Parameter (alpha5) angegeben, dann wird in dieser Variable der Text hinterlegt, der hinter dem Suchtext (alpha1) steht. Wird der Suchtext nicht gefunden, ist die Variable nach dem Aufruf leer. Die Länge des Nachfolgenden Textes richtet sich nach der Dimension der Variable. Bei einem alpha(10) beispielsweise, werden maximal 10 Zeichen zurückgegeben.

Beispiele:

```
// Suchen nach ganzem Wort mit Groß-/KleinschreibungtPos # tPrtRtf->PrtRtfSearch('Suche', _PrtRtf
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) übergebene Deskriptor ist ungültig.

ErrValueInvalid Optionen (int2) ungültig

Kontakt

PrtUnit(int1, int2) :



float

Umwandeln in Einheit

int1 Wert in logischen Einheiten
 Zieleinheit der

int2 Umwandlung

Resultat float Umgewandelte Einheit
Verwandte Befehle,

Siehe

PrtUnitLog()

Randeinstellungen, Seitengrößen usw. werden in den Eigenschaften in einer logischen Standard-Einheit angegeben. Diese Standard-Einheit kann mit dem Befehl PrtUnit() in andere Einheiten umgerechnet werden. Der umzurechnende Wert wird in (int1), die Zieleinheit in (int2) übergeben.

In folgende Einheiten kann umgerechnet werden:

PrtUnitTwips Umwandlung in Twips

PrtUnitMillimetres Umwandlung in Millimeter

PrtUnitCentimetres Umwandlung in Zentimeter

PrtUnitPoints Umwandlung in Points

PrtUnitInches Umwandlung in Inches

Der Rückgabewert ist der Wert in der gewünschten Zieleinheit.

Beispiel:

```
tLeftMargin # tPage->ppMarginLeft;// Rand in ZentimeterntLeftMargin # PrtUnit(tLeftMargin, _PrtUn
```

Kontakt

PrtUnitLog(float1, int2) : int



Umwandlung in logische Einheiten float1

Wert in Quelleinheiten

int2 Quelleinheit

Resultat int Wert in logischen Einheiten

Siehe Verwandte Befehle, PrtUnit()

Mit diesem Befehl kann ein Wert in einer vorgegebenen Einheit in Logische Einheiten umgewandelt werden. Folgende Konstanten für die Quelleinheit (int2) sind möglich:

PrtUnitTwips Umwandlung von Twips

PrtUnitMillimetres Umwandlung von Millimeter

PrtUnitCentimetres Umwandlung von Zentimeter

PrtUnitPoints Umwandlung von Points

PrtUnitInches Umwandlung von Inches

Beispiel:

```
// Zentimeter -> logische EinheitentLeftMargin # PrtUnitLog(tLeftMargin, __PrtUnitCentimetres);
```

Kontakt

PrtDeviceOpen([alpha1,
int2]) : handle Drucker-
Device öffnen alpha1
Druckernname



Optionen
int2 PrtDeviceSystem System-Drucker
 laden
Resultat handle Deskriptor des
 PrintDevice-Objekts
Siehe Verwandte Befehle,
 PrtDeviceClose(), PrtJobClose()

Der Befehl öffnet ein PrintDevice-Objekt. Werden keine Parameter übergeben, wird die Ausgabeeinheit des Windows-Standard-Drucker geöffnet.

In (alpha1) kann der Name eines Windows-Druckertreibers angegeben werden. In diesem Fall muss in (int2) die Konstante PrtDeviceSystem übergeben werden. Es wird dann die entsprechende Ausgabeeinheit geöffnet.

Die Namen der installierten Druckertreiber können aus der Liste der Drucker ausgelesen werden, die beim Applikationsobjekt hinterlegt sind. (Eigenschaft PrinterList, siehe auch PrtInfo()).

Der Deskriptor der Ausgabeeinheit kann beim Befehl PrtJobClose() angegeben werden, um in der Druckvorschau einen bestimmten Drucker voreinzustellen oder um auf einen bestimmten Drucker auszugeben.

Beispiele:

```
// Standard-Drucker öffnetPrintDevice # PrtDeviceOpen() ;// PDFWriter öffnetPrintDevice # PrtDev
```

Kontakt

obj -> PrtDeviceClose() :



logic

Drucker-Device schließen

Drucker-Device

obj

Deskriptor

Resultat logic Erfolgsstatus

Verwandte

Siehe Befehle,

PrtDeviceOpen()

Mit diesem Befehl wird ein durch PrtDeviceOpen() geöffnetes Drucker-Device wieder geschlossen.

Der von PrtDeviceOpen() zurückgegebene Deskriptor wird in (obj) übergeben.

Beispiel:

```
// Standarddrucker öffnetPrintDevice # PrtDeviceOpen();...// Drucker schließtPrintDevice->PrtD
```

Kontakt

obj -> PrtDeviceRefresh() : int |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | <img alt="Document icon" data-bbox="29268 71 2938

Kontakt

obj -> PrtPrinterRefresh() : int                            <img alt="Printer icon" data-bbox="3185 71 3188

Kontakt

Befehle für das PpcObject

Liste der Befehle und Konstanten des Druckertreibers

Befehlsgruppen,

Siehe Befehlsliste,

Druckprozessor

Befehle

- PpcMakeAcrobatPdf
- PpcMakeEps
- PpcMakePdf
- PpcMakePreviewBmp
- PpcMakePreviewJpg
- PpcMakePreviewPng
- PpcMakePreviewTif
- PpcMakeTif
- PpcPrint

Kontakt

obj -> PpcMakeAcrobatPdf(alpha1[, alpha2])



: int

PDF-Dokument mit Adobe Distiller erstellen

obj PpcObject-Deskriptor

alpha1 Pfad- und Dateiname

der PDF-Datei

Einstellungen des

alpha2 PDF-Dokuments

(optional)

Resultat int Fehlerwert

Verwandte Befehle,

PpcObject,

Siehe PpcMakePdf(),

PpcMakeTif(),

PpcMakeEps()

Der Befehl erzeugt ein PDF-Dokument unter Verwendung des Adobe-Distillers und kann nur in Verbindung mit dem PDF-Druckertreiber, jedoch nicht mit dem TIFF-Druckertreiber verwendet werden. Die Eigenschaften des erzeugten PDF-Dokuments können über diese Eigenschaften gesetzt werden.

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

In (alpha1) muss der Name des zu erzeugenden PDF-Dokuments angegeben werden.

Falls bereits eine Datei mit dem Namen existiert, wird diese überschrieben.



Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System".

Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

Mit dem Argument (alpha2) kann ein Name einer Konvertierungseinstellung, der auch in den Druckeinstellungen für den Acrobat-Distiller definiert ist, angegeben werden.

ErrOk kein Fehler

ErrPpcDriver Falscher Druckertreiber (zum Beispiel TIFF)

ErrPpcAcrobat Fehler bei der PDF-Erstellung

Kontakt

obj ->

PpcMakeEps(alpha1[,



int2]) : int

EPS-Dokument erstellen

obj PpcObject-Deskriptor

alpha1 Pfad- und Dateiname

des Dokuments

int2 Auflösung in dpi

Resultat int Fehlerwert

Verwandte Befehle,

PpcObject,

Siehe PpcMakePdf(),

PpcMakeAcrobatPdf(),

PpcMakeTif()

Der Befehl erzeugt ein Encapsulated Postscript-Dokument unter Verwendung von Ghostscript und kann nur in Verbindung mit dem PDF-Druckertreiber, nicht jedoch mit dem TIFF-Druckertreiber verwendet werden.

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

In (alpha1) muss der Name des zu erzeugenden Dokuments angegeben werden. Falls bereits eine Datei mit dem Namen existiert, wird diese überschrieben.



Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System".

Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

Mit dem Argument (int2) wird die DPI-Auflösung des zu erstellenden Dokuments angegeben. Wird das Argument nicht angegeben oder ist es 0, wird ein Dokument erzeugt, welches dieselbe Auflösung hat wie die Ausgangsdatei. Der höchste zulässige Wert ist 3200 DPI. Je höher die Auflösung gewählt wird, desto grösser wird das erzeugte Dokument. Für die Bildschirmanzeige sind beispielsweise 72 DPI ausreichend.

ErrOk kein Fehler

ErrPpcFileCreate Fehler bei der Dokument-Erstellung

ErrPpcFileOpen Beim Öffnen des Dokuments ist ein Fehler aufgetreten.

ErrPpcFileRead Beim Lesen des Dokuments ist ein Fehler aufgetreten.

ErrPpcFileWrite Beim Schreiben des Dokuments ist ein Fehler aufgetreten.

Kontakt

obj -> PpcMakePdf(alpha1[, alpha2]) : int      PDF-

Dokument mit GhostScript erstellen

obj PpcObject-Deskriptor
alpha1 Pfad- und Dateiname

der erzeugten Datei
Einstellungen für das

alpha2 PDF-Dokument
(optional)

Resultat int Fehlerwert

Verwandte Befehle,
PpcObject,

Siehe PpcMakeAcrobatPdf(),
PpcMakeTif(),
PpcMakeEps()

Der Befehl erzeugt ein PDF-Dokument unter Verwendung von Ghostscript und kann nur in Verbindung mit dem PDF-Druckertreiber, nicht jedoch mit dem TIFF-Druckertreiber verwendet werden. Die Eigenschaften des erzeugten PDF-Dokuments können über diese Eigenschaften gesetzt werden.

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

In (alpha1) muss der Name des zu erzeugenden PDF-Dokuments angegeben werden.

Falls bereits eine Datei mit dem Namen existiert, wird diese überschrieben.



Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System". Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

Mit dem Argument (alpha2) können Optionen, die die Darstellungs-Qualität des PDF-Dokuments beeinflussen, angegeben werden. Folgende Konstanten können angegeben werden:

PpcPdfDefault - PDF-Ausgabe für unterschiedliche Anzeigemedien

PpcPdfScreen - Niedrigauflösendes PDF für die Bildschirmanzeige

PpcPdfEBook - Besserauflösendes PDF für die EBook-Anzeige

PpcPdfPrinter - Druckoptimierte PDF-Ausgabe

PpcPdfPrepress - Druckvorstufe optimierte PDF-Ausgabe

PpcPdfA - PDF/A-konformes Dokument erstellen

Während der Durchführung des Befehles sind die folgenden Eigenschaften von Bedeutung:

GsPasswordOwner, GsPasswordUser, GsPermissions und GsEncryption.

Wird die Option PpcPdfA angegeben, wird die Eigenschaft GsOptions nicht verwendet, statt dessen wird folgender Parameterstring übergeben:

-dPDFA;-dNOOUTERSAVE;-dUseCIEColor;-sProcessColorModel=DeviceCMYK. Da ein PDF/A-Dokument nicht verschlüsselt werden kann, werden zudem die Eigenschaften GsPasswordOwner, GsPasswordUser, GsPermissions und GsEncryption ignoriert.

Kontakt



Zum Erstellen eines PDF/A-Dokuments wird mindestens die Ghostscript Version 8.71 benötigt.

Folgende Fehlerwerte können zurückgegeben werden.

ErrOk PDF-Erstellung erfolgreich

ErrPpcDriver Falscher Druckertreiber (zum Beispiel TIFF)

ErrPpcGhostscript Fehler bei der PDF-Erstellung

Kontakt



**obj -> PpcMakePreviewBmp(alpha1,
point2[, int3]) : int**

Preview-Dateien im BMP-Format erstellen

obj PpcObject-Deskriptor

alpha1 Pfad- und Dateiname der erzeugten
Datei(en)

point2 Breite und Höhe des Bildes

Farbmodus (optional)

PpcColorDepthTrue True Color (16

Millionen

int3 Farben) =

default

PpcColorDepthMono Monochrom

(schwarz/weiß)

Resultat int Fehlerwert

Verwandte Befehle, PpcObject,
PpcMakePreviewJpg(),

Siehe

PpcMakePreviewPng(),
PpcMakePreviewTif()

Der Befehl erzeugt einzelne Preview-Dateien ausgewählter Seiten der Originaldatei im Windows Bitmap-Format.

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

Für jede Seite, die in das durch die Eigenschaft SelectPvw definierte

Auswahlkriterium fällt, wird eine Preview-Datei erstellt. Damit möglichst einfach Dateinamen für die einzelnen Preview-Dateien generiert werden können, ist im Argument (alpha1) die Angabe des '%d' Format-Spezifizierers möglich.

Beispiel:

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewBmp('preview_, PointMake(150, 75)) ;
```

'%d' wird im Dateiname der Preview-Datei durch die entsprechende Seitennummer ersetzt. Das Beispiel erzeugt Previews aller ungeraden Seiten: preview_1.bmp, preview_3.bmp, preview_5.bmp, ...

Es können auch mehrere Format-Anweisungen im Dateiname angegeben werden.

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewBmp('pvw_, PointMake(150, 75)) ;
```

pvw_1_1.bmp, pvw_3_3.bmp, pvw_5_5.bmp, ...

Soll das Prozentzeichen im Dateiname benutzt werden, wird dies durch Angabe eines doppelten Prozentzeichens ('%%') erreicht.

Weiterhin sind folgende Formatierungsangaben zulässig:

'%<n>d'

Kontakt

'<n>' ist eine Ziffer (1 ... 9), die die Anzahl der Stellen festlegt. Nicht vorhandene Stellen werden durch Leerzeichen aufgefüllt.
'%0<n>d' wie '%<n>d' nur das nicht mit Leerzeichen, sondern mit Nullen aufgefüllt wird.

Die Angabe des Formatspezifizierers im Dateiname ist optional. Ist eine Datei mit demselben Namen bereits vorhanden, wird diese überschrieben.

 Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System".
Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

In (point2) wird Breite und Höhe des zu erstellenden Images in Pixel angegeben. Ist eine der Ausdehnungen 0, wird das Bild unter Beibehaltung des Seitenverhältnisses skaliert. Sind beide Ausdehnungen auf 0 gesetzt, wird ein Vorschaubild in Originalgröße erzeugt. Beim Ausdruck über den PDF Druckertreiber ist die Auflösung auf 300 dpi begrenzt. (int3) bestimmt die Farbtiefe der Bitmap. Zulässig sind die Konstanten PpcColorDepthTrue für 16 Millionen Farben und PpcColorDepthMono für schwarz/weiß-Bilder.

Mögliche Rückgabewerte:

<u>ErrOk</u>	kein Fehler
<u>ErrPpcFileOpen</u>	Beim Öffnen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileCreate</u>	Beim Erzeugen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileRead</u>	Beim Lesen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileWrite</u>	Beim Schreiben einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcArgument</u>	Es wurden nicht die korrekten Argumente übergeben.

Kontakt



obj -> PpcMakePreviewJpg(alpha1,

point2[, int3]) : int

Preview-Dateien im JPEG-Format erstellen

obj PpcObject-Deskriptor

 Pfad- und Dateiname

alpha1 der erzeugten

 Datei(en)

 Breite und Höhe des

point2 Bildes

 JPEG-Qualität in

int3

 Prozent

Resultat int Fehlerwert

Verwandte Befehle,

PpcObject,

Siehe PpcMakePreviewBmp(),

PpcMakePreviewPng(),

PpcMakePreviewTif()

Der Befehl erzeugt einzelne Preview-Dateien ausgewählter Seiten der Originaldatei im JPEG-Format.

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

Für jede Seite, die in das durch die Eigenschaft SelectPvw definierte

Auswahlkriterium fällt, wird eine Preview-Datei erstellt. Damit möglichst einfach Dateinamen für die einzelnen Preview-Dateien generiert werden können, ist im Argument (alpha1) die Angabe des '%d' Format-Spezifizierers möglich.

Beispiel:

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewJpg('preview_%d.jpg', PointMake(150,
```

'%d' wird im Dateiname der Preview-Datei durch die entsprechende Seitennummer ersetzt. Das Beispiel erzeugt Previews aller ungeraden Seiten: preview_1.jpg, preview_3.jpg, preview_5.jpg, ...

Es können auch mehrere Format-Anweisungen im Dateinamen angegeben werden.

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewJpg('pvw_%d_%d.jpg', PointMake(150, 7
```

Es werden dann folgende Dateien erzeugt: pvw_1_1.jpg, pvw_3_3.jpg, pvw_5_5.jpg, ...

Soll das Prozentzeichen im Dateiname benutzt werden, wird dies durch Angabe eines doppelten Prozentzeichens ('%%') erreicht.

Weiterhin sind folgende Formatierungsangaben zulässig:

'%<n>d' '<n>' ist eine Ziffer (1 ... 9), die die Anzahl der Stellen festlegt. Nicht vorhandene Stellen werden durch Leerzeichen aufgefüllt.

'%0<n>d'

Kontakt

wie '%<n>d' nur das nicht mit Leerzeichen, sondern mit Nullen aufgefüllt wird.

Die Angabe des Formatspezifizierers im Dateiname ist optional. Ist eine Datei mit demselben Namen bereits vorhanden, wird diese überschrieben.



Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System".

Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

In (point2) wird Breite und Höhe des zu erstellenden Images in Pixel angegeben. Ist eine der Ausdehnungen 0, wird das Bild unter Beibehaltung des Seitenverhältnisses skaliert. Sind beide Ausdehnungen auf 0 gesetzt, wird ein Vorschaubild in Originalgröße erzeugt. Beim Ausdruck über den PDF Druckertreiber ist die Auflösung auf 300 DPI begrenzt. (int3) bestimmt die Qualität des JPEGs. Zulässig sind die Werte 1 bis 100 (Prozent). Fehlt die Angabe ist die Qualität automatisch 100 Prozent.

Mögliche Rückgabewerte:

<u>ErrOk</u>	kein Fehler
<u>ErrPpcFileOpen</u>	Beim Öffnen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileCreate</u>	Beim Erzeugen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileRead</u>	Beim Lesen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileWrite</u>	Beim Schreiben einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcArgument</u>	Es wurden nicht die korrekten Argumente übergeben.

Kontakt

**obj -> PpcMakePreviewPng(alpha1,
point2) : int**



Preview-Dateien im PNG-Format erstellen

obj PpcObject-Deskriptor

Pfad- und Dateiname

alpha1 der erzeugten

Datei(en)

Breite und Höhe des

point2 Bildes

Resultat int Fehlerwert

Verwandte Befehle,

PpcObject,

Siehe PpcMakePreviewBmp(),

PpcMakePreviewJpg(),

PpcMakePreviewTif()

Der Befehl erzeugt einzelne Preview-Dateien ausgewählter Seiten der Originaldatei im PNG-Format (Portable Network Graphics).

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

Für jede Seite, die in das durch die Eigenschaft SelectPvw definierte Auswahlkriterium fällt, wird eine Preview-Datei erstellt. Damit möglichst einfach Dateinamen für die einzelnen Preview-Dateien generiert werden können, ist im Argument (alpha1) die Angabe des '%d' Format-Spezifizierers möglich.

Beispiel:

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewPng('preview_%d.png', PointMake(150,
```

'%d' wird im Dateiname der Preview-Datei durch die entsprechende Seitennummer ersetzt. Das Beispiel erzeugt Previews aller ungeraden Seiten: preview_1.png, preview_3.png, preview_5.png, ...

Es können auch mehrere Format-Anweisungen im Dateiname angegeben werden.

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewPng('p vw_%d_%d.png', PointMake(150, 7
```

Es werden dann folgende Dateien erzeugt: p vw_1_1.png, p vw_3_3.png, p vw_5_5.png, ...

Soll das Prozentzeichen im Dateiname benutzt werden, wird dies durch Angabe eines doppelten Prozentzeichens ('%%') erreicht.

Weiterhin sind folgende Formatierungssangaben zulässig:

'%<n>d' '<n>' ist eine Ziffer (1 ... 9), die die Anzahl der Stellen festlegt. Nicht vorhandene Stellen werden durch Leerzeichen aufgefüllt.

'%0<n>d' wie '%<n>d' nur das nicht mit Leerzeichen, sondern mit Nullen aufgefüllt wird.

Kontakt

Die Angabe des Formatspezifizierers im Dateiname ist optional. Ist eine Datei mit demselben Namen bereits vorhanden, wird diese überschrieben.



Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System". Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

In (point2) wird Breite und Höhe des zu erstellenden Images in Pixel angegeben. Ist eine der Ausdehnungen 0, wird das Bild unter Beibehaltung des Seitenverhältnisses skaliert. Sind beide Ausdehnungen auf 0 gesetzt, wird ein Vorschaubild in Originalgröße erzeugt. Beim Ausdruck über den PDF Druckertreiber ist die Auflösung auf 300 DPI begrenzt.

Mögliche Rückgabewerte:

<u>ErrOk</u>	kein Fehler
<u>ErrPpcFileOpen</u>	Beim Öffnen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileCreate</u>	Beim Erzeugen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileRead</u>	Beim Lesen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileWrite</u>	Beim Schreiben einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcArgument</u>	Es wurden nicht die korrekten Argumente übergeben.

Kontakt

obj -> PpcMakePreviewTif(alpha1, point2[,
int3]) : int



Preview-Dateien im TIFF-Format erstellen

obj PpcObject-Deskriptor

alpha1 Pfad- und Dateiname der erzeugten
Datei(en)

point2 Breite und Höhe des Bildes

Farbmodus (optional)

PpcColorDepthTrue True Color (16

Millionen

int3 Farben) =

default

PpcColorDepthMono Monochrom

(schwarz/weiß)

Resultat int Fehlerwert

Verwandte Befehle, PpcObject,
PpcMakePreviewBmp(),

Siehe

PpcMakePreviewJpg(),
PpcMakePreviewPng()

Der Befehl erzeugt einzelne Preview-Dateien ausgewählter Seiten der Originaldatei im TIFF-Format (Tagged Image File Format).

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

Für jede Seite, die in das durch die Eigenschaft SelectPvw definierte Auswahlkriterium fällt, wird eine Preview-Datei erstellt. Damit möglichst einfach Dateinamen für die einzelnen Preview-Dateien generiert werden können, ist im Argument (alpha1) die Angabe des '%d' Format-Spezifizierers möglich.

Beispiel:

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewTif('preview_%d.tif', PointMake(150,
```

'%d' wird im Dateinamen der Preview-Datei durch die entsprechende Seitennummer ersetzt. Das Beispiel erzeugt Previews aller ungeraden Seiten: preview_1.tif, preview_3.tif, preview_5.tif, ...

Es können auch mehrere Format-Anweisungen im Dateiname angegeben werden.

```
tPpcObject->ppSelectPvw # 'odd()' ; tPpcObject->PpcMakePreviewTif('pvw_%d_%d.tif', PointMake(150, 7
```

Es werden dann folgende Dateien erzeugt: pvw_1_1.tif, pvw_3_3.tif, pvw_5_5.tif, ...

Soll das Prozentzeichen im Dateiname benutzt werden, wird dies durch Angabe eines doppelten Prozentzeichens ('%%') erreicht.

Weiterhin sind folgende Formatierungsangaben zulässig:

'%<n>d'

Kontakt

'<n>' ist eine Ziffer (1 ... 9), die die Anzahl der Stellen festlegt. Nicht vorhandene Stellen werden durch Leerzeichen aufgefüllt.

'%0<n>d' wie '%<n>d' nur das nicht mit Leerzeichen, sondern mit Nullen aufgefüllt wird.

Die Angabe des Formatspezifizierers im Dateiname ist optional. Ist eine Datei mit demselben Namen bereits vorhanden, wird diese überschrieben.

 Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System". Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

In (point2) wird Breite und Höhe des zu erstellenden Images in Pixel angegeben. Ist eine der Ausdehnungen 0, wird das Bild unter Beibehaltung des Seitenverhältnisses skaliert. Sind beide Ausdehnungen auf 0 gesetzt, wird ein Vorschaubild in Originalgröße erzeugt. Beim Ausdruck über den PDF Druckertreiber ist die Auflösung auf 300 DPI begrenzt. (int3) bestimmt die Farbtiefe der TIFF-Datei. Zulässig sind die Konstanten PpcColorDepthTrue für 16 Millionen Farben und PpcColorDepthMono für schwarz/weiß-Bilder.

Mögliche Rückgabewerte:

<u>ErrOk</u>	kein Fehler
<u>ErrPpcFileOpen</u>	Beim Öffnen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileCreate</u>	Beim Erzeugen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileRead</u>	Beim Lesen einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcFileWrite</u>	Beim Schreiben einer Vorschau-Datei ist ein Fehler aufgetreten.
<u>ErrPpcArgument</u>	Es wurden nicht die korrekten Argumente übergeben.

Kontakt

obj ->

PpcMakeTif(alpha1[,



int2]) : int

TIFF-Dokument erstellen

obj PpcObject-Deskriptor

alpha1 Pfad- und Dateiname

des Dokuments

int2 Auflösung in dpi

Resultat int Fehlerwert

Verwandte Befehle,

PpcObject,

Siehe PpcMakePdf(),

PpcMakeAcrobatPdf(),

PpcMakeEps()

Der Befehl erzeugt ein TIFF-Image und kann sowohl ausgehend vom PDF- als auch vom TIFF-Druckertreiber verwendet werden. Die TIFF-Datei enthält genauso viele Seiten, wie die Originaldatei.

In (obj) wird der Deskriptor des PpcObject angeben. Der Deskriptor wird der Funktion des Druckprozessors übergeben.

In (alpha1) muss der Name des zu erzeugenden TIFF-Images angegeben werden. Falls bereits eine Datei mit dem Namen existiert, wird diese überschrieben.



Der Druckprozessor läuft als Dienst. D. h. das Programm läuft im Benutzerkontext "System".

Dieser Benutzer hat in der Regel keinen Zugriff auf Netzwerkressourcen. In (alpha1) können also nur lokale Laufwerke angegeben werden.

Mit dem Argument (int2) wird die DPI-Auflösung des zu erstellenden TIFF-Images angegeben.

Wird das Argument nicht angegeben oder ist es Null, wird eine TIFF-Image erzeugt, welches dieselbe Auflösung hat wie die Ausgangsdatei. Der höchste zulässige Wert ist 3200 DPI. Je höher die Auflösung gewählt wird, desto grösser wird das erzeugte Image. Für die Bildschirmanzeige sind beispielsweise 72 DPI ausreichend.

Mögliche Rückgabewerte:

ErrOk kein Fehler

ErrPpcFileCreate Fehler bei der TIFF-Erstellung

ErrPpcFileOpen Beim Öffnen des TIFF-Dokuments ist ein Fehler aufgetreten.

ErrPpcFileRead Beim Lesen des TIFF-Dokuments ist ein Fehler aufgetreten.

ErrPpcFileWrite Beim Schreiben des TIFF-Dokuments ist ein Fehler aufgetreten.

Kontakt

obj -> PpcPrint(handle1) : int |  |  |  |  |  | Druckausgabe

an Drucker weiterleiten

obj PpcObject-Deskriptor

handle1 PrintDevice-Deskriptor

Resultat int Fehlerwert

Siehe Verwandte Befehle,

PpcObject, PrintDevice

Diese Anweisung druckt den in (obj) übergebenen Druckjob in ein zuvor mit der Anweisung PrtDeviceOpen() geöffnetes PrintDevice (handle1). In (obj) wird der Deskriptor des PpcObject angeben, der der Funktion des Druckprozessors übergeben wurde.

Beispiel: Druckweiterleitung auf den Standard-Drucker

```
tDevice # PrtDeviceOpen(); if (tDevice > 0) { tPpcObject->PpcPrint(tDevice); tDevice->PrtDeviceCl
```



Da der Druckprozessor als Dienst auf dem Rechner läuft (siehe CONZEPT 16-Druckprozessor - Funktionsweise) können mit dieser Anweisung nur Windows-Druckertreiber angesprochen werden, die lokal auf dem Rechner installiert sind. Im Netzwerk installierte Drucker können nur angesprochen werden, wenn der Dienst unter dem lokalen Benutzerkonto betrieben und der lokale Benutzer (nicht der Benutzer in der Domäne) entsprechende Rechte besitzt.

Mögliche Rückgabewerte:

ErrOk kein Fehler

ErrPpcPrint Beim Drucken ist ein Fehler aufgetreten.

Kontakt

Befehle und Konstanten für Fehlermeldungen

Liste der Befehle und Konstanten für Fehlermeldungen Siehe

Befehlsgruppen, Befehlsliste,

Befehle

- ErrCall
- ErrGet
- ErrIgnore
- ErrMapText
- ErrPos
- ErrSet
- ErrThrowProc
- ErrTryCatch
- ErrTryIgnore
- XmlError

Konstanten

- Allgemeine Fehlerkonstanten

- ◆ ErrAll
- ◆ ErrData
- ◆ ErrDecryption
- ◆ ErrGeneric
- ◆ ErrOk
- ◆ ErrRights
- ◆ ErrTerminated
- ◆ ErrTimeout

- Konstanten für Laufzeitfehler

- ◆ ErrArgumentsDiff
- ◆ ErrArrayIndex
- ◆ ErrCallOld
- ◆ ErrCnv
- ◆ ErrCodeMissing
- ◆ ErrCodeUnknown
- ◆ ErrDataspaceDiff
- ◆ ErrDataspaceFree
- ◆ ErrDeadLock
- ◆ ErrDecimal
- ◆ ErrDivisionByZero
- ◆ ErrFileInvalid
- ◆ ErrFldType
- ◆ ErrFrameDiffers
- ◆ ErrFrameNotFound
- ◆ ErrHdlInvalid
- ◆ ErrIllegalOp
- ◆ ErrLinkInvalid
- ◆ ErrMathArgument

Kontakt

- ◆ [ErrMemExhausted](#)
- ◆ [ErrNoArgument](#)
- ◆ [ErrNoFile](#)
- ◆ [ErrNoFld](#)
- ◆ [ErrNoGlobalInfo](#)
- ◆ [ErrNoKey](#)
- ◆ [ErrNoKeyFld](#)
- ◆ [ErrNoLink](#)
- ◆ [ErrNoLinkFld](#)
- ◆ [ErrNoProcInfo](#)
- ◆ [ErrNoSbr](#)
- ◆ [ErrNoSub](#)
- ◆ [ErrPropInvalid](#)
- ◆ [ErrSelSame](#)
- ◆ [ErrSelSortDiffer](#)
- ◆ [ErrSelValueSet](#)
- ◆ [ErrServerTerm](#)
- ◆ [ErrStackOverflow](#)
- ◆ [ErrStringOverflow](#)
- ◆ [ErrValueInvalid](#)
- ◆ [ErrValueOverflow](#)
- ◆ [ErrValueRange](#)

- Konstanten für Fehler bei Datensatzoperationen

- ◆ [rDeadlock](#)
- ◆ [rExists](#)
- ◆ [rFailed](#)
- ◆ [rLastRec](#)
- ◆ [rLimitReached](#)
- ◆ [rLocked](#)
- ◆ [rMultiKey](#)
- ◆ [rNoKey](#)
- ◆ [rNoLock](#)
- ◆ [rNoRec](#)
- ◆ [rNoRights](#)
- ◆ [rOk](#)
- ◆ [rUserBreak](#)

- Konstanten für Datenbankfehler

- ◆ [ErrDbaAreaInUse](#)
- ◆ [ErrDbaAreaLocked](#)
- ◆ [ErrDbaAreaLockedAdmin](#)
- ◆ [ErrDbaAreaLockedDown](#)
- ◆ [ErrDbaAreaLockedNoStandbyOpen](#)
- ◆ [ErrDbaAreaLockedOpen](#)
- ◆ [ErrDbaAreaLockedOperation](#)
- ◆ [ErrDbaAreaLockedRollback](#)
- ◆ [ErrDbaAreaLockedStandby](#)
- ◆ [ErrDbaAreaOpen](#)

Kontakt

- ◆ ErrDbaAreaOperationDenied
- ◆ ErrDbaAreaPassword
- ◆ ErrDbaAreaRollback
- ◆ ErrDbaAreaStandby
- ◆ ErrDbaAreaType
- ◆ ErrDbaComm
- ◆ ErrDbaNoArea
- ◆ ErrDbaNoServer
- ◆ ErrDbaServerRelease
- ◆ ErrDbaServerStart
- ◆ ErrDbaUserInvalid
- ◆ ErrDbaUserLimit
- ◆ ErrDbaUserSelf

- Konstanten für Verarbeitungsfehler

- ◆ ErrAccessMode
- ◆ ErrEndOfData
- ◆ ErrExists
- ◆ ErrInProgress
- ◆ ErrInUse
- ◆ ErrKilled
- ◆ ErrLimitExceeded
- ◆ ErrLocked
- ◆ ErrMemIVInvalid
- ◆ ErrMemIVLength
- ◆ ErrMemKeyInvalid
- ◆ ErrMemKeyLength
- ◆ ErrMemMsgVerify
- ◆ ErrMemSgnInvalid
- ◆ ErrNameInvalid
- ◆ ErrOutOfMemory
- ◆ ErrRange
- ◆ ErrSvcSessionState
- ◆ ErrSystem
- ◆ ErrType
- ◆ ErrUnavailable
- ◆ ErrUnchangeable
- ◆ ErrUnerasable
- ◆ ErrUnknown

- Konstanten für Netzwerkinformationsfehler

- ◆ ErrNetCreate
- ◆ ErrNetIcmpID
- ◆ ErrNetIcmpType
- ◆ ErrNetNoHost
- ◆ ErrNetRead
- ◆ ErrNetReadLess
- ◆ ErrNetSelect
- ◆ ErrNetWrite

- Konstanten für Socketfehler

- ◆ ErrSckBindFailed
- ◆ ErrSckConnectFailed
- ◆ ErrSckCreate
- ◆ ErrSckDown
- ◆ ErrSckHostUnknown
- ◆ ErrSckNoLib
- ◆ ErrSckProxyAuthFailed
- ◆ ErrSckProxyConnectFailed
- ◆ ErrSckProxyFailed
- ◆ ErrSckProxyRead
- ◆ ErrSckProxyRefused
- ◆ ErrSckProxyUnknown
- ◆ ErrSckProxyWrite
- ◆ ErrSckRead
- ◆ ErrSckReadOverflow
- ◆ ErrSckSelect
- ◆ ErrSckTlsCertificateVerify
- ◆ ErrSckTlsConnect
- ◆ ErrSckWrite

- Konstanten für Fehler bei der Mehrfachselektion

- ◆ ErrMsdExists
- ◆ ErrMsdNotFound

- Konstanten für Fehler bei dynamischen Selektionen

- ◆ ErrParserEndOfText
- ◆ ErrParserIllegalElement
- ◆ ErrParserInvalidChar
- ◆ ErrParserInvalidConst
- ◆ ErrParserMissingComma
- ◆ ErrParserMissingParenthesis
- ◆ ErrParserOutOfRange
- ◆ ErrParserStringOverflow
- ◆ ErrParserSyntax
- ◆ ErrParserUnknownID
- ◆ ErrParserWrongType
- ◆ ErrSelCodeOverflow
- ◆ ErrSelIllegalOperator
- ◆ ErrSelInvalidField
- ◆ ErrSelInvalidKey
- ◆ ErrSelNoQuery
- ◆ ErrSelQueryOverflow
- ◆ ErrSelResultSet
- ◆ ErrSelTableOverflow
- ◆ ErrSelUnknownField
- ◆ ErrSelUnknownOrInvalidLink

Kontakt

- Konstanten für Fehler bei Storage-Objekten

- ◆ ErrStoInvalidFormat
- ◆ ErrStoLocked
- ◆ ErrStoNameInvalid
- ◆ ErrStoNoData
- ◆ ErrStoNoFile
- ◆ ErrStoNoPath
- ◆ ErrStoOperation

- Konstanten für Fehler der Tapi-Schnittstelle

- ◆ ErrTapiBadAddr
- ◆ ErrTapiBusy
- ◆ ErrTapiCallID
- ◆ ErrTapiCallState
- ◆ ErrTapiDevName
- ◆ ErrTapiDialString
- ◆ ErrTapiDialTimeout
- ◆ ErrTapiFailed
- ◆ ErrTapiInstall
- ◆ ErrTapiInUse
- ◆ ErrTapiMediaMode
- ◆ ErrTapiMemory
- ◆ ErrTapiNoConnect
- ◆ ErrTapiNoListen
- ◆ ErrTapiNotOwner
- ◆ ErrTapiReinit
- ◆ ErrTapiReject
- ◆ ErrTapiUnavail
- ◆ ErrTapiUnknown
- ◆ ErrTapiVersion

- Konstanten für Dateibearbeitungsfehler (extern)

- ◆ ErrFsiAccessDenied
- ◆ ErrFsiCurrentDir
- ◆ ErrFsiDriveInvalid
- ◆ ErrFsiExists
- ◆ ErrFsiHdlInvalid
- ◆ ErrFsiInvalidFormat
- ◆ ErrFsiLockViolation
- ◆ ErrFsiNoFile
- ◆ ErrFsiNoPath
- ◆ ErrFsiOpenFailed
- ◆ ErrFsiOpenOverflow
- ◆ ErrFsiOther
- ◆ ErrFsiReadFault
- ◆ ErrFsiSharingViolation
- ◆ ErrFsiWriteFault

Kontakt

- Konstanten für Fehler des OEM-Kits

- ◆ ErrOemDbaLock
- ◆ ErrOemInvalidFormat
- ◆ ErrOemOpenDesigner
- ◆ ErrOemOpenFailed
- ◆ ErrOemOpenFrame
- ◆ ErrOemOutOfSpace
- ◆ ErrOemPassword

- Konstanten für Fehler der binären Objekte

- ◆ ErrBinData
- ◆ ErrBinDecryption
- ◆ ErrBinDirNotEmpty
- ◆ ErrBinExists
- ◆ ErrBinLocked
- ◆ ErrBinNameInvalid
- ◆ ErrBinNoData
- ◆ ErrBinNoFile
- ◆ ErrBinNoLock
- ◆ ErrBinNoPath
- ◆ ErrBinOperation

- Konstanten für Fehler der Validierungsbefehle

- ◆ ErrVldExists
- ◆ ErrVldLocked
- ◆ ErrVldNameInvalid
- ◆ ErrVldNoFile
- ◆ ErrVldNoLock

- Konstanten für Fehler des Druckprozessors

- ◆ ErrPpcAcrobat
- ◆ ErrPpcArgument
- ◆ ErrPpcDriver
- ◆ ErrPpcFileCreate
- ◆ ErrPpcFileOpen
- ◆ ErrPpcFileRead
- ◆ ErrPpcFileWrite
- ◆ ErrPpcGhostscript
- ◆ ErrPpcPrint

- Konstanten für Fehler bei der Verarbeitung von PDF-Dokumenten

- ◆ ErrPdfImageFormat
- ◆ ErrPdfInsertMetaFile
- ◆ ErrPdfNotLicensed
- ◆ ErrPdfNotPdfA

Kontakt

- ◆ [ErrPdfPageAppend](#)
- ◆ [ErrPdfPageClosed](#)
- ◆ [ErrPdfPageNotExisting](#)
- ◆ [ErrPdfPassword](#)

- Konstanten für Fehler bei der Verwendung von regulären Ausdrücken

- ◆ [ErrRegExBadEscapeSequence](#)
- ◆ [ErrRegExBadInterval](#)
- ◆ [ErrRegExInvalidBackRef](#)
- ◆ [ErrRegExInvalidFlag](#)
- ◆ [ErrRegExInvalidRange](#)
- ◆ [ErrRegExLookBehindLimit](#)
- ◆ [ErrRegExMaxLtMin](#)
- ◆ [ErrRegExMismatchedParentheses](#)
- ◆ [ErrRegExMissingCloseBracket](#)
- ◆ [ErrRegExNotSupported](#)
- ◆ [ErrRegExNumberTooBig](#)
- ◆ [ErrRegExOctalTooBig](#)
- ◆ [ErrRegExPropertySyntax](#)
- ◆ [ErrRegExRuleSyntax](#)
- ◆ [ErrRegExSetContainsString](#)
- ◆ [ErrRegExStackOverflow](#)
- ◆ [ErrRegExTimeout](#)

- Konstanten für Fehler der ODBC-Schnittstelle

- ◆ [ErrOdbcEnvironment](#)
- ◆ [ErrOdbcError](#)
- ◆ [ErrOdbcFunctionFailed](#)
- ◆ [ErrOdbcIncomplete](#)
- ◆ [ErrOdbcNoData](#)
- ◆ [ErrOdbcNotFound](#)
- ◆ [ErrOdbcWarning](#)

- Konstanten für Fehler der Benutzerverwaltung

- ◆ [ErrUrmObjectNotFound](#)
- ◆ [ErrUrmParentNotFound](#)

- Konstanten für Fehler bei XML-Befehlen

- ◆ [ErrXmlFatal](#)
- ◆ [ErrXmlNotValid](#)
- ◆ [ErrXmlRecoverable](#)
- ◆ [ErrXmlWarning](#)

- Weitere Fehlerkonstanten

- ◆ [ErrPrtPaperFormat](#)
- ◆ [ErrRtfSyntaxError](#)

ErrCall(alpha1)

Prozedur bei Laufzeitfehlern

Verwandte

Befehle, try,

Laufzeitfehler

In dieser Funktion kann in (alpha1) eine Prozedur oder Funktion angegeben werden, die bei Laufzeitfehlern vor der entsprechenden Bildschirmmeldung aufgerufen wird. Eine Funktion innerhalb einer Prozedur kann nach dem Prozedurnamen mit einem : getrennt angegeben werden.

Beispiel:

```
ErrCall('Err:ErrorHandler');
```

Bei einem Fehler wird die Funktion ErrorHandler in der Prozedur Err aufgerufen.

Der Funktion können keine Parameter übergeben werden. Um Informationen über den Laufzeitfehler zu erhalten, stehen die folgenden Eigenschaften des System-Objektes (Sys) zur Verfügung:

- ErrCode - Fehlerwert des Laufzeitfehlers
- ErrText - Text der Fehlermeldung
- ErrProc - Prozedurfunktion, in der der Fehler aufgetreten ist
- ErrLine - Programmzeile, in der der Fehler aufgetreten ist
- ErrSource - Name der Include-Prozedur

Es kann ein Rückgabewert vom Typ logic definiert werden. Wird in diesem Fall true zurückgegeben, verhält sich die Funktion wie ohne definierten Rückgabewert. Es wird eine Laufzeitfehler-Meldung von CONZEPT 16 generiert. Wird false oder nichts zurückgegeben, erscheint kein Laufzeitfehler.



Falls im SOA-Service die Funktion ein false zurückliefert, wird der Laufzeitfehler nicht protokolliert. Bei aktiver Debugerverbindung erfolgt jedoch immer eine Anzeige des Fehlers im Debugger, unabhängig vom Rückgabewert.



Die Funktion, die zum Fehler führte, wird unabhängig vom Rückgabewert abgebrochen.

Beispiele:

```
// Anzeige des CONZEPT 16-Laufzeitfehlers nach der Funktionsub ErrorHandler{ ... } // odersub ErrHan
```

Die Prozedur wird mit dem Befehl ErrCall() bestimmt und bleibt so lange erhalten, bis sie durch einen erneuten Aufruf von ErrCall() ersetzt oder durch die Übergabe eines Leerstrings " entfernt wird.

Der Laufzeitfehler wird nach dem Durchführen der Prozedur angezeigt. In dieser Prozedur kann bei einem Laufzeitfehler ein Protokoll geschrieben werden, bevor die fehlerhafte Prozedur abgebrochen wird. Um einen Laufzeitfehler zu unterdrücken und in der eigenen Programmierung zu verarbeiten, muss der Befehl try verwendet werden.

Kontakt

Tritt während der Verarbeitung der ErrCall()-Funktion ein Laufzeitfehler auf, bricht die Funktion ohne eine Fehlermeldung ab.

Kontakt

ErrGet() : int         Globalen Fehlerwert
abfragen

Resultat int Fehlerwert

Verwandte

Siehe Befehle, try,
ErrCall()

Mit dieser Funktion kann der aktuelle globale Fehlerwert ermittelt werden. Im Debugger kann das Überwachen des globalen Fehlerwertes auf der Seite System vorgenommen werden.

Die im folgenden aufgeführten Fehlerkategorien dienen als Anhaltspunkt, bei welchen Befehlen die entsprechenden Fehler auftreten können. Die Fehlerwerte werden von den Befehlen an die laufende Prozedur zurückgegeben. Lediglich die Fehler aus der Kategorie "Laufzeitfehler" führen zu einem Abbruch der laufenden Prozedur, es sei denn, sie werden durch einen try-Befehl verarbeitet.

Folgende Fehlerwerte sind definiert:

Datensatzoperationen

Code Konstanten

- 0 rOK (Kein Fehler)
- 1 rLocked
- 2 rMultiKey
- 3 rNoKey
- 4 rLastRec
- 5 rNoRec
- 6 rExists
- 7 rNoLock
- 8 rUserBreak
- 9 rNoRights
- 10 rDeadlock
- 11 rLimitReached
- 12 rFailed

Allgemeiner Fehler

Code Konstante

- 0 ErrOk (Kein Fehler) -1
- ErrGeneric
- 2 ErrTimeout -5
- ErrData
- 6 ErrDecryption -7
- ErrRights
- 9 ErrTerminated

Verarbeitungsfehler

Code	Konstante
-12	<u>ErrOutOfMemory</u>
-51	<u>ErrRange</u>
-52	<u>ErrType</u>
-53	<u>ErrSystem</u>
-55	<u>ErrInProgress</u>
-56	<u>ErrNameInvalid</u>
-57	<u>ErrAccessMode</u>
-58	<u>ErrExists</u>
-59	<u>ErrLocked</u>
-60	<u>ErrLimitExceeded</u>
-61	<u>ErrUnerasable</u>
-62	<u>ErrUnchangeable</u>
-64	<u>ErrUnavailable</u>
-65	<u>ErrUnknown</u>
-66	<u>ErrKilled</u>
-67	<u>ErrInUse</u>
-68	<u>ErrEndOfData</u>
-69	<u>ErrMemKeyInvalid</u>
-70	<u>ErrMemSgnInvalid</u>
-71	<u>ErrMemMsgVerify</u>
-72	<u>ErrMemKeyLength</u>
-73	<u>ErrMemIVLength</u>
-74	<u>ErrMemIVInvalid</u>
-75	<u>ErrMemDecrypt</u>
-2920	<u>ErrSvcSessionState</u>

Externe Dateioperationen

Code	Konstante
-20	<u>ErrFsiNoFile</u>
-21	<u>ErrFsiNoPath</u>
-22	<u>ErrFsiOpenOverflow</u>
-23	<u>ErrFsiAccessDenied</u>
-24	<u>ErrFsiHdlInvalid</u>
-25	<u>ErrFsiDriveInvalid</u>
-26	<u>ErrFsiCurrentDir</u>
-27	<u>ErrFsiSharingViolation</u>
-28	<u>ErrFsiLockViolation</u>
-29	<u>ErrFsiOpenFailed</u>
-31	<u>ErrFsiReadFault</u>
-32	<u>ErrFsiWriteFault</u>
-35	<u>ErrFsiInvalidFormat</u>

-39 [ErrFsiOther](#)

-40 [ErrFsiExists](#)

Laufzeitfehler

Code Konstante

-160 [ErrStackOverflow](#)

-161 [ErrCodeMissing](#)

-162 [ErrCodeUnknown](#)

-164 [ErrMemExhausted](#)

-169 [ErrCallOld](#)

-170 [ErrNoProcInfo](#)

-171 [ErrNoGlobalInfo](#)

-172 [ErrDataSpaceDiff](#)

-173 [ErrDataSpaceFree](#)

-174 [ErrNoSub](#)

-175 [ErrArgumentsDiff](#)

-176 [ErrNoFld](#)

-177 [ErrFldType](#)

-178 [ErrArrayIndex](#)

-179 [ErrValueOverflow](#)

-180 [ErrStringOverflow](#)

-181 [ErrDivisionByZero](#)

-182 [ErrMathArgument](#)

-183 [ErrValueRange](#)

-184 [ErrNoFile](#)

-185 [ErrNoSbr](#)

-186 [ErrNoKey](#)

-187 [ErrNoLink](#)

-188 [ErrValueInvalid](#)

-189 [ErrNoKeyFld](#)

-190 [ErrNoLinkFld](#)

-191 [ErrHdlInvalid](#)

-192 [ErrNoArgument](#)

-193 [ErrLinkInvalid](#)

-194 [ErrFileInvalid](#)

-195 [ErrSelValueSet](#)

-196 [ErrSelSortDiffer](#)

-197 [ErrSelSame](#)

-199 [ErrPropInvalid](#)

-200 [ErrDecimal](#)

-201 [ErrCnv](#)

-202 [ErrFrameDiffers](#)

-203 [ErrFrameNotFound](#)

-205 [ErrIllegalOp](#)

-206 [ErrDeadLock](#)

Datenbankfehler

Code Konstante

-801 [ErrDbaNoServer](#)

-802 [ErrDbaComm](#)

-803 [ErrDbaNoArea](#)

-804 [ErrDbaAreaOpen](#)

-805 [ErrDbaAreaLocked](#)

-806 [ErrDbaAreaInUse](#)

-807 [ErrDbaAreaType](#)

-808 [ErrDbaAreaPassword](#)

-809 [ErrDbaUserLimit](#)

-810 [ErrDbaServerStart](#)

-811 [ErrDbaUserInvalid](#)

-813 [ErrDbaUserSelf](#)

-817 [ErrDbaAreaRollback](#)

-827 [ErrDbaAreaStandby](#)

-830 [ErrDbaAreaLockedAdmin](#)

-831 [ErrDbaAreaLockedOperation](#)

-832 [ErrDbaAreaLockedDown](#)

-833 [ErrDbaAreaLockedStandby](#)

-834 [ErrDbaAreaLockedRollback](#)

-835 [ErrDbaAreaLockedOpen](#)

-836 [ErrDbaAreaLockedNoStandbyOpen](#)

Socketfehler

Code Konstante

-701 [ErrSckNoLib](#)

-705 [ErrSckHostUnknown](#)

-706 [ErrSckCreate](#)

-707 [ErrSckConnectFailed](#)

-709 [ErrSckSelect](#)

-710 [ErrSckRead](#)

-711 [ErrSckReadOverflow](#)

-713 [ErrSckWrite](#)

-714 [ErrSckBindFailed](#)

-717 [ErrSckDown](#)

-722 [ErrSckProxyUnknown](#)

-724 [ErrSckProxyRefused](#)

-725	<u>ErrSckProxyConnectFailed</u>
-726	<u>ErrSckProxyRead</u>
-727	<u>ErrSckProxyWrite</u>
-732	<u>ErrSckProxyFailed</u>
-733	<u>ErrSckProxyAuthFailed</u>
-741	<u>ErrSckTlsConnect</u>
-745	<u>ErrSckTlsCertificateVerify</u>

Netzwerkinformations-Fehler

Code Konstante

-705	<u>ErrNetNoHost</u>
-706	<u>ErrNetCreate</u>
-709	<u>ErrNetSelect</u>
-710	<u>ErrNetRead</u>
-712	<u>ErrNetReadLess</u>
-713	<u>ErrNetWrite</u>
-719	<u>ErrNetIcmpType</u>
-720	<u>ErrNetIcmpID</u>

Fehler von binären Objekten

Code Konstante

-1501	<u>ErrBinNameInvalid</u>
-1502	<u>ErrBinNoPath</u>
-1503	<u>ErrBinNoFile</u>
-1504	<u>ErrBinNoData</u>
-1505	<u>ErrBinNoLock</u>
-1506	<u>ErrBinLocked</u>
-1507	<u>ErrBinExists</u>
-1508	<u>ErrBinDirNotEmpty</u>
-1509	<u>ErrBinData</u>
-1510	<u>ErrBinOperation</u>
-1512	<u>ErrBinDecryption</u>

Fehler von Storage-Objekten

Code Konstante

-1501	<u>ErrStoNameInvalid</u>
-1502	<u>ErrStoNoPath</u>
-1503	<u>ErrStoNoFile</u>
-1504	<u>ErrStoNoData</u>
-1506	<u>ErrStoLocked</u>
-1510	<u>ErrStoOperation</u>
-1511	<u>ErrStoInvalidFormat</u>

Fehler von Validierungsbefehlen

Code	Konstante
-1501	<u>ErrVldNameInvalid</u>
-1503	<u>ErrVldNoFile</u>
-1505	<u>ErrVldNoLock</u>
-1506	<u>ErrVldLocked</u>
-1507	<u>ErrVldExists</u>

TAPI-Fehler

Code	Konstante
-1801	<u>ErrTapiUnknown</u>
-1802	<u>ErrTapiVersion</u>
-1803	<u>ErrTapiDevName</u>
-1804	<u>ErrTapiInUse</u>
-1805	<u>ErrTapiDialString</u>
-1806	<u>ErrTapiDialTimeout</u>
-1807	<u>ErrTapiInstall</u>
-1808	<u>ErrTapiReinit</u>
-1809	<u>ErrTapiMemory</u>
-1810	<u>ErrTapiFailed</u>
-1811	<u>ErrTapiUnavail</u>
-1812	<u>ErrTapiMediaMode</u>
-1813	<u>ErrTapiBusy</u>
-1814	<u>ErrTapiBadAddr</u>
-1815	<u>ErrTapiNoConnect</u>
-1816	<u>ErrTapiReject</u>
-1817	<u>ErrTapiCallState</u>
-1819	<u>ErrTapiCallId</u>
-1820	<u>ErrTapiNotOwner</u>
-1821	<u>ErrTapiNoListen</u>

Druckprozessor

Code	Konstante
-9074	<u>ErrPpcAcrobat</u>
-9034	<u>ErrPpcArgument</u>
-9070	<u>ErrPpcDriver</u>
-9054	<u>ErrPpcFileCreate</u>
-9051	<u>ErrPpcFileOpen</u>
-9052	<u>ErrPpcFileRead</u>
-9053	<u>ErrPpcFileWrite</u>
-9015	<u>ErrPpcGhostscript</u>
-9081	<u>ErrPpcPrint</u>

Fehler von dynamischen Selektionen

Code Konstante

- 2100 [ErrParserEndOfText](#)
- 2101 [ErrParserInvalidChar](#)
- 2102 [ErrParserInvalidConst](#)
- 2103 [ErrParserWrongType](#)
- 2104 [ErrParserOutOfRange](#)
- 2105 [ErrParserStringOverflow](#)
- 2106 [ErrParserUnknownID](#)
- 2107 [ErrParserSyntax](#)
- 2108 [ErrParserIllegalElement](#)
- 2109 [ErrParserMissingParenthesis](#)
- 2110 [ErrParserMissingComma](#)
- 2201 [ErrSelUnknownField](#)
- 2202 [ErrSelInvalidField](#)
- 2203 [ErrSelUnknownOrInvalidLink](#)
- 2204 [ErrSelIllegalOperator](#)
- 2205 [ErrSelQueryOverflow](#)
- 2206 [ErrSelResultSet](#)
- 2207 [ErrSelTableOverflow](#)
- 2208 [ErrSelCodeOverflow](#)
- 2209 [ErrSelNoQuery](#)
- 2210 [ErrSelInvalidKey](#)

Fehler bei der Verarbeitung von PDF-Dokumenten

Code Konstante

- 2501 [ErrPdfImageFormat](#)
- 2502 [ErrPdfPassword](#)
- 2503 [ErrPdfPageClosed](#)
- 2504 [ErrPdfPageNotExisting](#)
- 2505 [ErrPdfPageAppend](#)
- 2506 [ErrPdfInsertMetafile](#)
- 2507 [ErrPdfNotPdfA](#)
- 2508 [ErrPdfNotLicensed](#)

Fehler bei der Verwendung von regulären Ausdrücken

Code Konstante

- 2702 [ErrRegExRuleSyntax](#)
- 2703 [ErrRegExBadEscapeSequence](#)
- 2704 [ErrRegExPropertySyntax](#)
- 2705 [ErrRegExNotSupported](#)
- 2706 [ErrRegExMismatchedParentheses](#)
- 2707 [ErrRegExNumberTooBig](#)

- 2708 [ErrRegExBadInterval](#)
- 2709 [ErrRegExMaxLtMin](#)
- 2710 [ErrRegExInvalidBackRef](#)
- 2711 [ErrRegExInvalidFlag](#)
- 2712 [ErrRegExLookBehindLimit](#)
- 2713 [ErrRegExSetContainsString](#)
- 2714 [ErrRegExOctalTooBig](#)
- 2715 [ErrRegExMissingCloseBracket](#)
- 2716 [ErrRegExInvalidRange](#)
- 2717 [ErrRegExStackOverflow](#)
- 2718 [ErrRegExTimeout](#)

Fehler der ODBC-Schnittstelle

Code Konstante

- 551 [ErrOdbcNotFound](#)
- 552 [ErrOdbcIncomplete](#)
- 553 [ErrOdbcEnvironment](#)
- 554 [ErrOdbcFunctionFailed](#)
- 555 [ErrOdbcError](#)
- 556 [ErrOdbcWarning](#)
- 557 [ErrOdbcNoData](#)

Fehler der Benutzerverwaltung

Code Konstante

- 2301 [ErrUrmObjectNotFound](#)
- 2302 [ErrUrmParentNotFound](#)

Fehler bei XML-Befehlen

Code Konstante

- 2401 [ErrXmlWarning](#)
- 2402 [ErrXmlRecoverable](#)
- 2403 [ErrXmlFatal](#)
- 2404 [ErrXmlNotValid](#)

Fehler bei der Mehrfachselektion

Code Konstante

- 1750 [ErrMsdExists](#)
- 1751 [ErrMsdNotFound](#)

Fehler des OEM-Kits

Code Konstante

- 1501 [ErrOemPassword](#)
- 1502 [ErrOemDbaLock](#)

Kontakt

- 1503 [ErrOemOpenFailed](#)
- 1504 [ErrOemInvalidFormat](#)
- 1505 [ErrOemOutOfSpace](#)
- 1506 [ErrOemOpenDesigner](#)
- 1507 [ErrOemOpenFrame](#)

Weitere Fehler

Code Konstante

- 1601 [ErrPrtPaperFormat](#)
- 1701 [ErrRtfSyntaxError](#)

Eigene Fehlerwerte

Sofern eigene Fehlerwerte benutzt werden, sollten diese Werte von -10000 oder niedriger besitzen. Der allgemeine Fehlerwert kann mit [ErrSet\(\)](#) gesetzt werden.

Kontakt

ErrIgnore(int1, logic2)



Zu ignorierende Laufzeitfehler festlegen

Fehlerwert

[ErrCallOld](#)

Fehler beim Aufruf von

A- Prozeduren

Fehler bei

int1

Typkonvertierung

[ErrDecimal](#)

Fehler bei

Dezimalberechnung

[ErrStringOverflow](#) Zeichenkettenüberlauf

logic2 Laufzeitfehlermeldung ausschalten (true)

oder einschalten (false)

Siehe [Verwandte Befehle](#)

Mit diesem Befehl kann die Generierung eines Laufzeitfehlers abgeschaltet werden.

- [ErrCallOld](#)

Wird eine A- Prozedur in einer Umgebung aufgerufen, die keine A- Prozeduren verarbeitet, wird ein Laufzeitfehler generiert. Mit dieser Option wird der Laufzeitfehler unterdrückt. Die Prozedur wird nicht aufgerufen.

- [ErrCnv](#)

Der Laufzeitfehler bei einer nicht durchführbaren Typkonvertierung wird unterdrückt.

- [ErrDecimal](#)

Kommt es bei einer Berechnung mit Dezimalzahlen zu einem Fehler, wird der Laufzeitfehler unterdrückt. Als Ergebnis wird der Wert [DecimalError](#) zurückgegeben.

- [ErrStringOverflow](#)

Anstatt eines Laufzeitfehlers wird bei einer Zuweisung einer zu langen Zeichenkette der Alphawert abgeschnitten.

Weitere Laufzeitfehler können nur in einem try-Block ignoriert werden

([ErrTryIgnore\(\)](#)).



Es ist zu beachten, dass das Abschalten von Laufzeitfehlern die Fehlersuche erschweren kann.

Kontakt

**ErrMapText(int1[,
alpha2[, int3]]) : alpha**



Fehlertext ermitteln int1

Fehlerwert

alpha2 Sprachkennung (optional)

FehlerTyp (optional)

_ErrMapC16 CONZEPT

16-Fehlerwert

int3

_ErrMapSys Windows-Fehlerwert

_ErrMapX509 Fehlerwert der

Zertifikatsüberprüfung

Resultat alpha

FehlerText

Siehe Verwandte Befehle, ErrGet()

Diese Funktion ermittelt aus dem in (int1) übergebenen Fehlerwert den dazugehörigen FehlerText. Es können alle Fehlerwerte von Laufzeit- und Übersetzungsfehlern, sowie Windows-Fehlerwerte übergeben werden. Der Fehlerwert wird entweder durch den entsprechenden Befehl zurückgegeben oder kann mit dem Befehl ErrGet() bzw. über die Eigenschaft ErrCode ermittelt werden.

In (alpha2) kann eine Sprachkennung angegeben werden. Der FehlerText wird in der entsprechenden Sprache zurückgegeben. Folgende Sprachkennungen stehen zur Verfügung:

'DE' deutscher FehlerText

'EN' englischer FehlerText (default)

'*U' FehlerText in der Systemsprache des aktuellen Windows-Benutzers

Die Groß- und Kleinschreibung wird nicht unterschieden. Wird keine Sprachkennung angegeben, wird der englische FehlerText zurückgegeben. Kann der übergebene Fehlerwert nicht in einen Text umgewandelt werden, wird ein Leerstring zurückgegeben.

Im Parameter (int3) kann die Art des Fehlerwertes angegeben werden. Dazu stehen folgende Konstanten zur Verfügung:

- **_ErrMapC16**

In (int1) wurde ein CONZEPT 16-Fehlerwert angegeben.

- **_ErrMapSys**

In (int1) wurde ein Windows-Fehlerwert angegeben.

- **_ErrMapX509**

In (int1) wurde ein Fehlerwert einer Zertifikatsüberprüfung angegeben (siehe SckOptVerify).

Beispiele:

```
tErg # ProcCompile(tProcedure);if (tErg != _ErrOk){  WinDialogBox(tHdlParent, 'Error compiling pr
```

Kontakt

Mögliche Laufzeitfehler

ErrValueInvalid In (alpha2) wurden eine unbekannte Sprachkennung angegeben.

Kontakt

ErrPos() : int

Fehlerposition abfragen

Resultat **int Label**

Verwandte

Siehe

Befehle

Mit dieser Funktion kann die Stelle des aufgetretenen Fehlers ermittelt werden, in dem das Resultat mit den definierten Labels verglichen wird. Das Label muss zuvor mit :<name> definiert werden.

Ein Label kann nur innerhalb eines try-Blocks verwendet werden.

Ein Beispiel zur Verwendung von Labels befindet sich in der Beschreibung der Anweisung try.

Kontakt

ErrSet(int1)  **Globalen**

Fehlerwert setzen

int1 Fehlerwert

Verwandte

Siehe [Befehle](#),
[ErrGet\(\)](#)

Bei (int1) gleich [ErrOk](#) wird der globale Fehlerwert gelöscht. Mit diesem Befehl können ebenfalls eigene Fehlerwerte gesetzt werden. Nach dem Setzen durch ErrSet() wird sofort der [try](#)-Block verlassen und der entsprechende Fehlerwert kann ausgewertet werden.

Kontakt

ErrThrowProc()

**Fehler auslösende Funktion
Prozedur- und
Resultat alpha**

Funktionsname

Siehe **Verwandte Befehle, trysub, try**

Wird ein **try**- oder **trysub**-Block durch einen Fehler verlassen, kann mit dieser Funktion der Prozedur- und Funktionsname ermittelt werden, in der der Fehler ausgelöst wurde. Die Anweisung gibt die Namen in der Form <Prozedur>:<Funktion> zurück.

Der Wert bleibt solange erhalten, bis ein neuer **try**- oder **trysub**-Block beginnt.

Beispiel:

```
sub myFunc{ try { ... ErrSet(_ErrGeneric); ... }main{ try { myFunc(); } if (E
```

Kontakt

ErrTryCatch(int1, logic2) |  Behandlung eines

Laufzeitfehlers ändern

int1 Fehlerwert

Fehler abfangen

logic2 (true) oder nicht
abfangen (false)

Verwandte

Befehle, try,

Siehe ErrTryIgnore(),
Fehlerbehandlung
(Blog)

Mit dieser Anweisung können Laufzeitfehler innerhalb von try-Blöcken abgefangen werden. Dazu wird vor dem try-Block der Befehl mit dem entsprechenden Laufzeitfehler (int1) und (logic2 = true) aufgerufen. Die Behandlung durch die Fehlerbehandlungsroutine des Programmierers kann durch die Übergabe von (logic2 = false) wieder aufgehoben werden.

Wird in (int1) ErrAll angegeben, wird für alle Laufzeitfehler die Fehlerbehandlung durch den Programmierer gesetzt oder zurückgesetzt.

Üblicherweise führen Laufzeitfehler zu einer Fehlermeldung und dem Abbruch der laufenden Funktion, sie können aber durch die Verwendung von ErrTryCatch() nach einem try-Block durch die Applikation verarbeitet werden.

Ein Beispiel für die Programmierung befindet sich in der Beschreibung des try-Blocks.

ErrTryIgnore(int1[, int2]) |  Zu ignorierende

Fehlerwerte setzen

int1 Fehlerwert

Ende des

int2 Fehlerwertebereiches
(optional)

Verwandte Befehle,
try, ErrTryCatch(),

Siehe

Fehlerbehandlung

Mit dieser Anweisung können Fehler, die normalerweise zum Verlassen eines try-Blocks führen, innerhalb des try-Blocks verarbeitet werden. Die Anweisung ErrTryIgnore() wird selbst in dem entsprechenden try-Block aufgerufen und ist gültig, bis der try-Block beendet wird. Befindet sich innerhalb des Blocks ein weiterer try-Block, muss die Anweisung für diesen Block separat angegeben werden.

Mit ErrTryIgnore() können alle Fehler angegeben werden. Bei Verwendung von ErrAll werden alle Fehler inklusive Laufzeitfehler innerhalb des try-Blocks ignoriert. Bei ErrOk wird kein Fehler ignoriert.



Wurde ErrAll angegeben und zusätzlich mit ErrTryCatch() die Behandlung eines Laufzeitfehlers gesetzt, wird ErrTryCatch() bevorzugt. Tritt der Laufzeitfehler auf, wird der try-Block verlassen.

Durch die Angabe eines zweiten Fehlerwertes in (int2) werden alle Fehler in dem angegebenen Bereich ignoriert. Dabei ist es nicht relevant, ob zuerst die untere oder die obere Grenze des Bereiches angegeben wird.

Beispiel:

```
try{ // Datensatzfehler im TRY-Block behandeln ErrTryIgnore(_rLocked, _rDeadlock); ... }switch
```

Ein ausführlicheres Beispiel befindet sich bei der Beschreibung der Anweisung try.

Kontakt

XmlError(int1) : alpha  **Weitere Informationen zu einer XML-Fehler ermitteln**

int1 Information, die ermittelt werden soll Resultat

alpha Zeichenkette mit Informationen Siehe

Verwandte Befehle, Blog

Tritt beim Laden einer XML-Datei (**XmlLoad()**) ein Fehler auf, können mit dieser Anweisung weitere Informationen über diesen Fehler ermittelt werden. Folgende Informationen stehen über einen XML-Fehler zur Verfügung:

- **_XmlErrorText (0)**

Es wird der Fehlertext zurückgegeben.

- **_XmlErrorCode (1)**

Es wird der Fehlerwert zurückgegeben.

- **_XmlErrorLine (2)**

Es wird die Zeile, in der der Fehler aufgetreten ist, zurückgegeben.

- **_XmlErrorColumn (3)**

Es wird die Spalte, in der der Fehler aufgetreten ist, zurückgegeben.

Mögliche Laufzeitfehler

ErrValueInvalid In (int1) wurde ein ungültiger Wert übergeben.

Kontakt

Allgemeine Fehlerkonstanten

Siehe <u>Alle Befehle</u>	
<u>ErrAll</u>	Symbolischer Wert für alle Fehler
<u>ErrData</u>	Datenfehler
<u>ErrDecryption</u>	Fehler beim Entschlüsseln einer Datei
<u>ErrGeneric</u>	Allgemeiner Fehler ist aufgetreten
<u>ErrOk</u>	Kein Fehler aufgetreten
<u>ErrRights</u>	Keine ausreichenden Rechte
<u>ErrTerminated</u>	Job wurde ohne Fehler beendet
<u>ErrTimeout</u>	Zeitüberschreitung ist aufgetreten

Kontakt

_ErrAll

Symbolischer Wert für alle Fehler bei [ErrTryIgnore\(\)](#)
-2.147.483.647

Wert / **0x80000001**

Siehe [Verwandte Befehle](#)

Option von [ErrTryIgnore\(\)](#) - Es werden alle Fehler ignoriert.

Kontakt

_ErrData

Datenfehler

Wert -5

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Allgemein

Ursache : Beim Verarbeiten der Daten ist ein allgemeiner Fehler aufgetreten.

Kontakt

_ErrDecryption

Fehler beim Entschlüsseln einer Datei

Wert -6

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Allgemein

Ursache : Die Datei konnte mit dem angegebenen Schlüssel nicht entschlüsselt werden

Bei dem Befehl [FsiFileProcess\(\)](#) wird dieser Fehler zurückgegeben, wenn die externe Datei mit dem angegebenen Schlüssel nicht entschlüsselt werden konnte.

Kontakt

_ErrGeneric

Ein allgemeiner Fehler ist aufgetreten

Wert -1

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Allgemein

Ursache : Ist das Resultat des Befehls [DbaKeyRebuild\(\)](#) _ErrGeneric, sind bei der Reorganisation Kollisionen bei Schlüsselwerten aufgetreten, oder es existieren Datensätze ohne Schlüssel.

Bei anderen Funktionen ist bei diesem Fehler kein genauer Fehlertyp spezifiziert.

Kontakt

_ErrOk

Kein Fehler aufgetreten

Wert 0

Verwandte

Siehe **Befehle**,

ErrGet()

Die Operation ist ohne Fehler durchgeführt worden.

Kontakt

_ErrRights

Keine ausreichenden Rechte

Wert -7

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Allgemein

Ursache : Der Benutzer verfügt nicht über ausreichende Rechte, um die entsprechende Funktion durchzuführen.

Kontakt

_ErrTerminated

Job wurde ohne Fehler beendet

Wert -9

Verwandte

Siehe Befehle,

JobErrorCode

Kategorie : Allgemein

Ursache : Hat die Eigenschaft JobErrorCode den Wert _ErrTerminated wurde der entsprechende Job ohne einen Fehler beendet.

Kontakt

_ErrTimeout

Beim Lesen oder Schreiben des Sockets ist ein Timeout aufgetreten Wert -2

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Allgemein

Ursache : Es ist eine Zeitüberschreitung aufgetreten.

Kontakt

Konstanten für Laufzeitfehler

<u>Alle</u>	
Siehe <u>Befehle</u>	
<u>ErrArgumentsDiff</u>	Funktionsargumente stimmen nicht überein
<u>ErrArrayIndex</u>	Der Index für ein Array hat einen ungültigen Wert
<u>ErrCallOld</u>	A- Prozedur aufgerufen
<u>ErrCnv</u>	Fehler bei Typkonvertierung aufgetreten
<u>ErrCodeMissing</u>	Prozedurcode fehlt
<u>ErrCodeUnknown</u>	Prozedurcode fehlerhaft
<u>ErrDataspaceDiff</u>	Die Definition eines globalen Datenbereichs ist abweichend definiert
<u>ErrDataspaceFree</u>	Globaler Datenbereich ist nicht im Speicher angelegt
<u>ErrDecimal</u>	Fehler bei Dezimalberechnung
<u>ErrDivisionByZero</u>	Division durch Null
<u>ErrFileInvalid</u>	Datei ungültig
<u>ErrFldType</u>	Feldtyp abweichend
<u>ErrFrameDifters</u>	Frame abweichend
<u>ErrFrameNotFound</u>	Frame nicht vorhanden
<u>ErrHdlInvalid</u>	Ungültiger Deskriptor
<u>ErrIllegalOp</u>	Unzulässige Anweisung
<u>ErrLinkInvalid</u>	Verknüpfung ungültig
<u>ErrMathArgument</u>	Argument bei einer mathematischen Funktion unzulässig
<u>ErrMemExhausted</u>	Speicher konnte nicht angefordert werden
<u>ErrNoArgument</u>	Argument fehlt
<u>ErrNoFile</u>	Datei ist nicht in der Datenstruktur vorhanden
<u>ErrNoFld</u>	Feld nicht vorhanden
<u>ErrNoGlobalInfo</u>	Zu einem globalen Datenbereich wurde keine Information gefunden
<u>ErrNoKey</u>	Schlüssel ist nicht in der Datenstruktur vorhanden
<u>ErrNoKeyFld</u>	Schlüsselfeld ist nicht in der Datenstruktur vorhanden
<u>ErrNoLink</u>	Verknüpfung ist nicht in der Datenstruktur vorhanden
<u>ErrNoLinkFld</u>	Verknüpfungsfeld ist nicht in der Datenstruktur vorhanden
<u>ErrNoProcInfo</u>	Beim Start einer Prozedur wurde keine Prozedurinformation gefunden
<u>ErrNoSbr</u>	Teildatensatz ist nicht in der Datenstruktur vorhanden
<u>ErrNoSub</u>	Prozedurfunktion nicht vorhanden
<u>ErrPropInvalid</u>	Prozedurfunktion nicht vorhanden
<u>ErrSelSame</u>	Es wird mehrfach die gleiche Selektion angegeben
<u>ErrSelSortDiffer</u>	Sortierung der Selektion weicht ab
<u>ErrSelValueSet</u>	Kombination von zwei Selektionen mit Wertmenge
<u>ErrStackOverflow</u>	Stapelüberlauf
<u>ErrStringOverflow</u>	Alphanumerischer Wert zu lang
<u>ErrValueInvalid</u>	

Kontakt

Ungültiger Wert bei einer Typumwandlung oder bei einer Eingabeprüfung
Wert zu groß oder zu klein
Wert außerhalb des zulässigen Wertebereichs

ErrValueOverflow

ErrValueRange

_ErrArgumentsDiff

Funktionsargumente stimmen nicht überein Wert -

175

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Die an eine Funktion übergebenen Argumente weichen von den dort definierten Argumenten ab (Typ bzw. Anzahl der Argumente). Nach Anderung der Funktion wurden die abhängigen Prozeduren nicht neu übersetzt.
Der Fehler kann beim Aufruf einer Prozedurfunktion auftreten.

_ErrArrayIndex

Der Index für ein Array hat einen ungültigen Wert Wert -

178

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Der Wert des Indizes ist entweder kleiner 1 oder größer als die Anzahl der Elemente im Array.

Der Fehler kann bei der Benutzung einer Variablen eines Arrays auftreten.

Kontakt

_ErrCallOld

A- Prozedur aufgerufen

Wert -169

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#),

[CallOld\(\)](#)

Kategorie : Laufzeitfehler

Ursache : In einer Umgebung, in der keine A- Prozeduren verarbeitet werden können (zum Beispiel im Advanced-Client, Programmierschnittstelle usw.), wurde eine A-Prozedur aufgerufen.

Kontakt

_ErrCnv

Fehler bei Typkonvertierung aufgetreten

Wert -201

Verwandte Befehle,

Siehe Konvertierungsbefehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Ist eine Typkonvertierung aufgrund des Ausgangswerts nicht möglich, erfolgt der Laufzeitfehler "Typkonvertierung nicht möglich".



Mit dem Befehl ErrIgnore() kann die Generierung des Laufzeitfehlers abgeschaltet werden.
In diesem Fall wird der globale Fehlerwert auf _ErrCnv gesetzt.

Kontakt

_ErrCodeMissing

Prozedurcode fehlt

Wert -161

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Bei der Durchführung einer Prozedur ist der Code nicht vorhanden.

Kontakt

_ErrCodeUnknown

Prozedurcode fehlerhaft

Wert -162

Verwandte

Siehe [Befehle](#),

[SelRun\(\)](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Bei der Durchführung einer Prozedur ist eine Anweisung angegeben, die nicht ausgeführt werden kann (zum Beispiel [WinDialogBox\(\)](#) in einer Prozedur, die vom Server ausgeführt wird).

Kontakt

_ErrDataspaceDiff

Die Definition eines globalen Datenbereichs ist abweichend definiert Wert -172

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Nach Änderung der Definition eines globalen
Datenbereichs wurden die abhängigen Prozeduren nicht neu übersetzt.

Der Fehler kann beim Aufruf einer Prozedurfunktion auftreten.

Kontakt

_ErrDataspaceFree

Globaler Datenbereich ist nicht im Speicher angelegt Wert -

173

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Der globale Datenbereich wurde nicht mit VarAllocate() angelegt oder mit VarFree() wieder freigegeben.

Der Fehler kann bei der Benutzung einer globalen Variablen auftreten.

Kontakt

_ErrDeadLock

Deadlock aufgetreten

Wert -206

Verwandte

Befehle,

Siehe ErrGet(),

DeadLockRTE,

rDeadlock

Kategorie : Laufzeitfehler

Ursache : In der Eigenschaft Options des Sys-Objektes ist die Option
DeadLockRTE gesetzt. Weiterhin ist bei der Ausführung einer
Datensatzoperationen ein Deadlock (rDeadlock) aufgetreten.

_ErrDecimal

Fehler bei Dezimalberechnung

Wert -200

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#),

[decimal](#)

Kategorie : Laufzeitfehler

Ursache : Bei einem ungültigen Ergebnis einer Berechnung erfolgt der Laufzeitfehler "Fehler bei Dezimalberechnung".



Mit dem Befehl [ErrIgnore\(\)](#) kann die Generierung des Laufzeitfehlers abgeschaltet werden. In diesem Fall wird der globale Fehlerwert auf [_ErrDecimal](#) gesetzt.

Kontakt

_ErrDivisionByZero

Division durch Null

Wert -181

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Bei einer Division hat der Divisor den Wert 0.

Der Fehler kann bei den Divisionsoperatoren %, / oder div auftreten.

Kontakt

_ErrFileInvalid

Datei ungültig

Wert -194

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Die angegebene Datei kann bei der Operation nicht verwendet werden.

Kontakt

_ErrFldType

Feldtyp abweichend

Wert -177

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler / Funktionsresultat

Ursache : Das angegebene Feld hat in der Datenstruktur einen anderen Typ als erwartet.

Der Fehler kann bei der Benutzung eines Feldes der Datenstruktur oder der Zuweisung zu einer Variablen eines falschen Types auftreten.

Kontakt

_ErrFrameDiffers

Frame abweichend

Wert -202

Siehe [Verwandte Befehle](#),
[ErrGet\(\)](#),
[with](#)

Kategorie : Laufzeitfehler

Ursache : Ein Frame, welcher durch ein with-Statement referenziert wird, wurde nach dem Übersetzen der Prozedur geändert.

Kontakt

_ErrFrameNotFound

Frame nicht vorhanden

Wert -203

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#),

[with](#)

Kategorie : Laufzeitfehler

Ursache : Ein durch ein [with](#)-Statement referenzierter Frame ist nicht vorhanden.

_ErrHdlInvalid

Ungültiger Deskriptor

Wert -191

Verwandte

Siehe [Befehle](#),

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Es wurde ein Deskriptor angegeben, der entweder nicht definiert ist oder vom falschen Typ bzw. Untertyp ist.



Wird an einen ...Close()-Befehl als Deskriptor der Wert 0 oder ein Fehlerwert (< 0) übergeben, wird dieser Laufzeitfehler nicht erzeugt.

Über die Funktion [HdlInfo\(\)](#) kann der Typ eines Deskriptors ermittelt werden.

Kontakt

_ErrIllegalOp

Unzulässige Anweisung

Wert -205

Verwandte

Siehe

Befehle

Kategorie : Laufzeitfehler

Ursache : Es wurde eine Operation ausgeführt, die nicht zulässig ist.

Dieser Fehler wird zum Beispiel dann zurückgegeben, wenn einer Selektion, die bereits nach einem Schlüssel sortiert ist, ein Sortierfeld zugewiesen wird.

Kontakt

_ErrLinkInvalid

Verknüpfung ungültig

Wert -193

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : **Die angegebene Verknüpfung passt nicht zur Zielfile oder zum angegebenen Filter.**

Kontakt

[_ErrMathArgument](#)

Argument bei einer mathematischen Funktion unzulässig Wert -182

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Als Argument einer mathematischen Funktion wurde ein Wert angegeben, der unzulässig ist.

Der Fehler kann bei folgenden Funktionen auftreten:

- [LogN\(\)](#)
- [Log2\(\)](#)
- [Log10\(\)](#)
- [Sqrt\(\)](#)

Kontakt

_ErrMemExhausted

Speicher konnte nicht angefordert werden

Wert -164

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Der angeforderte Speicher konnte nicht allokiert werden.

_ErrNoArgument

Argument fehlt

Wert -192

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Bei einer Systemfunktion wurden weniger Argumente übergeben, als für die gewünschte Operation benötigt werden.

Kontakt

_ErrNoFile

Datei ist nicht in der Datenstruktur vorhanden Wert -

184

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Die angegebene Dateinummer ist nicht in der Datenstruktur vorhanden.

Kontakt

_ErrNoFld

Feld nicht vorhanden

Wert -176

Verwandte

Siehe [Befehle](#),

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Das angegebene Feld ist in der Datenstruktur nicht vorhanden.

Der Fehler kann bei der Benutzung eines Feldes der Datenstruktur auftreten.

Kontakt

_ErrNoGlobalInfo

Zu einem globalen Datenbereich wurde keine Information gefunden Wert -171

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Die Prozedur, in der der globale Datenbereich definiert ist, wurde gelöscht.

Der Fehler kann beim Aufruf einer Prozedurfunktion auftreten.

Kontakt

_ErrNoKey

Schlüssel ist nicht in der Datenstruktur vorhanden Wert -

186

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : **Das in dem Schlüssel angegebene Feld ist nicht in der Datenstruktur vorhanden.**

Kontakt

_ErrNoKeyFld

Schlüsselfeld ist nicht in der Datenstruktur vorhanden Wert -
189

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Das in dem Schlüssel angegebene Schlüsselfeld ist nicht in der

Ursache : Datenstruktur vorhanden.

Kontakt

_ErrNoLink

Verknüpfung ist nicht in der Datenstruktur vorhanden Wert -

187

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Die angegebene Verknüpfung ist nicht in der Datenstruktur vorhanden.

Kontakt

_ErrNoLinkFld

Verknüpfungsfeld ist nicht in der Datenstruktur vorhanden Wert -

190

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : **Das in der Verknüpfung angegebene Feld ist nicht in der Datenstruktur vorhanden.**

Kontakt

_ErrNoProcInfo

Beim Start einer Prozedur wurde keine Prozedurinformation gefunden Wert -170

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Die entsprechende Prozedur ist nicht vorhanden oder wurde nie übersetzt.

Der Fehler kann beim Aufruf einer Prozedurfunktion auftreten. Dieser Laufzeitfehler kann nicht innerhalb eines try-Blocks abgefangen werden. Durch den Aufruf der Prozedur (oder Funktion) wird der try-Block verlassen. Erst jetzt kann festgestellt werden, dass die Prozedur oder Funktion nicht vorhanden ist.

Der Fehler tritt ebenfalls auf, wenn innerhalb eines Clients, der nur A+ Prozeduren unterstützt mit der Anweisung CallOld() eine 4.0 kompatible Prozedur aufgerufen wird.

Kontakt

_ErrNoSbr

Teildatensatz ist nicht in der Datenstruktur vorhanden Wert -

185

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Der angegebene Teildatensatz ist nicht in der Datenstruktur vorhanden.

Kontakt

_ErrNoSub

Prozedurfunktion nicht vorhanden

Wert -174

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Die angegebene Prozedurfunktion ist in der Prozedur nicht vorhanden.

Der Fehler kann beim Aufruf einer Prozedurfunktion auftreten.

Kontakt

_ErrPropInvalid

Ungültige Eigenschaft

Wert -199

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Es wurde eine Eigenschaft angegeben, über die das Objekt nicht verfügt.

Kontakt

_ErrSelSame

Es wird mehrfach die gleiche Selektion angegeben Wert -

197

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Bei der Anweisung SelRun() sollen zwei identische Selektionen miteinander kombiniert werden.

Kontakt

[_ErrSelSortDiffer](#)

Sortierung der Selektion weicht ab

Wert -196

[Verwandte](#)

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler

Ursache : Es sollen zwei Selektionen miteinander
kombiniert werden, die unterschiedliche Sortierungen besitzen.

Kontakt

_ErrSelValueSet

Kombination von zwei Selektionen mit Wertmenge Wert -

195

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Es sollen zwei Selektionen mit SelUnion, SelInter oder SelMinus Ursache :
miteinander kombiniert werden, wobei eine von beiden Selektionen
eine Wertmenge enthält.

Kontakt

_ErrServerTerm

Server wurde beendet

Wert -166

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Während RmtCall() oder SelRun(..., SelServer...) wurde der Server

Ursache : beendet, die Datenbank geschlossen, oder der Benutzer <Intern> abgemeldet
wurde.

Kontakt

_ErrStackOverflow

Stapelüberlauf

Wert -160

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Bei der Durchführung einer Prozedur kam es zu einem Stapelüberlauf. Dazu kann es beispielsweise kommen, wenn eine Funktion zu oft rekursiv aufgerufen wird. Die maximale Aufruftiefe von Funktionen beträgt 160 Verschachtelungen.

_ErrStringOverflow

Alphanumerischer Wert zu lang

Wert -180

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Der angegebene Alphawert ist länger als zulässig.

Überschreitet die Länge der Zeichenkette die Definitionsgrenze von 65.520 Zeichen, wird der Fehler ErrValueRange erzeugt.



Mit dem Befehl ErrIgnore() kann die Generierung des Laufzeitfehlers

_ErrStringOverflow bei der Zuweisung von Alphawerten abgeschaltet werden. Anstatt eines Laufzeitfehlers wird bei einer Zuweisung einer zu langen Zeichenkette der Alphawert abgeschnitten.

Kontakt

_ErrValueInvalid

Ungültiger Wert bei einer Typumwandlung oder bei einer Eingabeprüfung Wert -188

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Laufzeitfehler / Funktionsresultat

Ursache : Bei einer Typumwandlung oder bei einer Eingabeprüfung ist ein ungültiger Wert aufgetreten.

Kontakt

_ErrValueOverflow

Wert zu groß oder zu klein

Wert -179

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Der benutzte Wert liegt außerhalb des Wertebereichs des Zieltyps.

Der Fehler kann an folgenden Stellen auftreten:

- Zuweisung an ein Feld bzw. eine Variable vom Typ byte, word oder int
- Funktion StrChar()

_ErrValueRange

Wert außerhalb des zulässigen Wertebereichs Wert -

183

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Laufzeitfehler

Der benutzte Wert liegt außerhalb des für die Funktion gültigen Wertebereichs. Der Fehler wird auch dann erzeugt, wenn eine

Ursache : **Zeichenkette mit mehr als 65.520 Zeichen einer Funktion übergeben oder** einer Variablen zugewiesen werden soll.

Der Fehler **ErrStringOverflow** wird ausgelöst, wenn das Ziel der Zuweisung für die zugewiesene Anzahl von Zeichen nicht ausreicht. Überschreitet die Länge der Zeichenkette die Definitionsgrenze, wird dieser Fehler erzeugt.

Der Fehler kann bei folgenden Funktionen auftreten:

- **Rnd()**
- **StrIns()**
- **StrChar()**
- **StrDecrypt()**
- **StrEncrypt()**
- **DbaControl()**
- **DbaConnect()**
- **DbaDisconnect()**
- **WseInfo()**

Kontakt

Konstanten für Fehler bei Datensatzoperationen

Siehe [Alle Befehle](#)

<u>rDeadlock</u>	Es ist eine Verklemmung aufgetreten
<u>rExists</u>	Datensatz existiert bereits
<u>rFailed</u>	Zugriff auf temporären Baum nicht möglich
<u>rLastRec</u>	Schlüssel nicht vorhanden / Letzter Datensatz
<u>rLimitReached</u>	Datensatzlimit überschritten
<u>rLocked</u>	Datensatz ist gesperrt
<u>rMultiKey</u>	Schlüssel ist nicht eindeutig
<u>rNoKey</u>	Schlüssel nicht vorhanden / Kein leerer Schlüssel vorhanden
<u>rNoLock</u>	Datensatz ist nicht gesperrt
<u>rNoRec</u>	Kein Datensatz vorhanden
<u>rNoRights</u>	Benutzerrechte nicht ausreichend
<u>rOk</u>	Operation erfolgreich
<u>rUserBreak</u>	Operation abgebrochen

Kontakt

_rDeadlock

Es ist eine Verklemmung aufgetreten

Wert 10

Verwandte

Befehle,

Siehe DtaBegin(),

DtaCommit(),

DtaRollback(),

ErrGet()

Wird dieses Resultat zurückgegeben, konnte die Funktion nicht ausgeführt werden, da innerhalb einer Transaktion eine Verklemmung aufgetreten ist.

Eine Verklemmung tritt dann auf, wenn innerhalb einer Transaktion auf Datenbankelemente zugegriffen wird, deren Segmente gerade von einer anderen Transaktion verwendet werden. Die Transaktion geht in einen Wartezustand bis die entsprechenden Segmente von der anderen Transaktion wieder freigegeben werden.

Benötigt die andere Transaktion allerdings Segmente von Datenbankelementen, die von der ersten Transaktion bereits verändert wurden, kommt es zu einem Deadlock. Beide Transaktionen gehen in einen Wartezustand und warten auf die andere Transaktion.

Ein solcher Deadlock wird von CONZEPT 16 erkannt. Das System bricht eine der Transaktionen ab (vgl. DtaRollback()). Der Befehl, der die Verklemmung ausgelöst hat, gibt den Wert _rDeadlock zurück. Die Prozedur muss diesen Wert in einer entsprechenden Fehlerbehandlung verarbeiten.

Wird in der Eigenschaft Options des Sys-Objektes die Option DeadLockRTE gesetzt, wird statt der Rückgabe von _rDeadlock der Laufzeitfehler ErrDeadLock ausgelöst.

Beispiel:

```
try{ DtaBegin(); ... RecRead(); ... DtaCommit(); }switch (ErrGet()){ ... case _rDeadlock :
```



Änderungen an den folgenden Datenbankinhalten werden von der Transaktion berücksichtigt:

- Datensätze
- Prozeduren
- Texte
- Binäre Verzeichnisse und Objekte
- Dialog-Objekte
- Bilder in der Ressourcenverwaltung
- Selektionen

Kontakt

_rExists

Datensatz existiert bereits

Wert 6

Verwandte

Befehle,

Siehe RecInsert(),

RecReplace(),

SelRecInsert(),

ErrGet()

Kategorie : Datensatzoperation

Der Datensatz konnte nicht eingefügt oder rückgespeichert werden, da Ursache :
bereits ein Satz mit einem identischen eindeutigen Schlüsselwert
existiert.

Kontakt

_rFailed

Zugriff auf temporären Baum nicht möglich Wert

12

Verwandte

Befehle,

RecRead(),

Siehe RecLink(),

SelRecInsert(),

SelRecDelete(),

ErrGet()

Kategorie : Datensatzoperation

Es wird auf einen temporären Baum zugegriffen, der nicht mehr

Ursachen : vorhanden ist. Temporäre Bäume werden beispielsweise für Selektionsmengen und für Dateien mit gesetztem Haken bei temporäre Datei angelegt.

Ist die Client-Version kleiner als 5.8.11, wird stattdessen der Fehlercode _rNoRec zurückgegeben.

Kontakt

_rLastRec

Schlüssel nicht vorhanden / Letzter Datensatz Wert 4

Verwandte

Befehle,

OEMSave(),

Siehe RecRead(),

RecDelete(),

SelRead(),

ErrGet()

Kategorie : Datensatzoperation

Ursache : In der Datei ist kein Satz mit dem gewünschten Schlüsselwert und auch kein Satz mit einem größeren Schlüsselwert vorhanden.
Es wurde der Satz mit dem größten Schlüsselwert geladen.

Kontakt

_rLimitReached

Datensatzlimit überschritten

Wert 11

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Datensatzoperation

**Ursache : Die maximale Anzahl der zu ermittelnden
Datensätze wurde überschritten.**

Die Hauptgebnismenge enthält trotz des Fehlerwertes die gefundenen Datensätze bis zum Limit.

Kontakt

_rLocked

Datensatz ist gesperrt

Wert 1

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Datensatzoperation

Ursache : Der gesuchte Datensatz ist vorhanden und von einem anderen Benutzer gesperrt.

Bei RecRead() wurde der Satz geladen, sofern nicht die Option RecNoLoad verwendet wurde.

Liefert der Befehl RecRead() das Ergebnis _rOk, kann mit einem weiteren RecRead() mit der Option RecCheckLock festgestellt werden, ob eine gemeinsame Sperre eingerichtet wurde.

Der sperrende Benutzer kann mit dem Befehl UserInfo(_UserLocked) ermittelt werden.

Wird der Wert von einer RmtData...()-Anweisung zurückgegeben, konnte das Datenobjekt nicht gesperrt oder überschrieben werden, weil es von einem anderen Benutzer gesperrt ist.

Kontakt

_rMultiKey

Schlüssel ist nicht eindeutig

Wert 2

Verwandte

Siehe Befehle,

RecRead(),

ErrGet()

Kategorie : Datensatzoperation

Ursache :

In der Datei sind mehrere Sätze mit dem gewünschten Schlüsselwert vorhanden.

Es wurde der erste der Sätze mit dem nicht eindeutigen Schlüsselwert geladen.

Kontakt

_rNoKey

Schlüssel nicht vorhanden / Kein leerer Schlüssel vorhanden Wert 3

Verwandte

Befehle,

RecRead(),

Siehe SelRead(),

SelDelete(),

DbaKeyRebuild(),

ErrGet()

Kategorie : Datensatzoperation

Ursache : In der Datei ist kein Satz mit dem gewünschten Schlüsselwert vorhanden.

Es wurde der Satz mit dem nächst größeren Schlüsselwert geladen.

Die Funktion DbaKeyRebuild() gibt als Resultat _rNoKey zurück, wenn die Funktion mit der Option KeyOnlyEmpty gestartet wurde und kein leerer Schlüssel vorhanden ist.

Die Befehle UserPassword(), UserCreate() und UserDelete() geben den Fehler zurück, wenn ein unbekannter Benutzer angegeben wurde.

_rNoLock

Datensatz ist nicht gesperrt

Wert 7

Verwandte

Befehle,

RecReplace(),

Siehe **SelRecInsert()**,

SelRecDelete(),

SelRun(),

ErrGet()

Kategorie : Datensatzoperation

Ursache : Der Datensatz konnte nicht zurückgespeichert werden, da er in der Datenbank nicht gesperrt ist.

Das Resultat wird auch bei Selektionsoperationen benutzt, wenn die Selektion nicht gesperrt ist.

Kontakt

_rNoRec

Kein Datensatz vorhanden

Wert 5

Verwandte

Befehle,

RecRead(),

Siehe RecLink(),

SelRead(),

SelRecDelete(),

SelRun(),

ErrGet()

Kategorie : Datensatzoperation

- Es wurde kein Satz geladen, da entweder die Datei leer ist oder kein vorhergehender bzw. nachfolgender Satz existiert.

Ursachen :

Datensatz.

- Bei einem Zugriff über eine Verknüpfung existiert kein verknüpfter

Wird der Wert von einer RmtDataRead()-Anweisung zurückgegeben, konnte das Datenobjekt nicht gefunden werden.

Kontakt

_rNoRights

Benutzerrechte nicht ausreichend

Wert 9

[Verwandte](#)

[Befehle](#),

Siehe [Textbefehle](#),

[Datensatzbefehle](#),

[ErrGet\(\)](#)

Kategorie : Datensatzoperation

Ursache :

Wird dieses Resultat zurückgegeben, sind die Berechtigungen des Benutzers für die Operation nicht ausreichend.

Die Rechte für [Textbefehle](#) können mit dem Befehl [TextInfo\(\)](#) überprüft werden.

Kontakt

_rOk

Operation erfolgreich

Wert 0

Verwandte

Siehe Befehle,

ErrGet()

Dieses Resultat hat je nach Befehl folgende Bedeutung:

<u>RecRead()</u>	Der Satz mit dem angeforderten Schlüssel wurde geladen (Je nach verwendeten Optionen kann die Bedeutung abweichend sein).
<u>RecInsert()</u>	Der Satz wurde eingefügt.
<u>RecReplace()</u>	Der Satz wurde zurückgespeichert.
<u>RecDelete()</u>	Der Satz wurde gelöscht.
<u>RecLink()</u>	Verknüpfter Datensatz wurde gelesen.
<u>SelClear()</u>	Selektion wurde geleert.
<u>SelRead()</u>	Selektion wurde gelesen.
<u>SelRun()</u>	Selektion durchgeführt.
<u>SysExecute()</u>	Programm konnte erfolgreich gestartet werden.
<u>TextRead()</u>	Text/Prozedur wurde gelesen.
<u>TextDelete()</u>	Text/Prozedur wurde gelöscht.
<u>TextCreate()</u>	Text/Prozedur wurde angelegt.
<u>TextCopy()</u>	Text/Prozedur wurde kopiert.
<u>TextRename()</u>	Text/Prozedur wurde umbenannt.
<u>WinAdd()</u>	Objekt konnte erfolgreich dem Parent-Objekt angehängen werden.

Kontakt

_rUserBreak

Operation abgebrochen

Wert 8

Verwandte

Befehle,

Siehe DbaKeyRebuild(),

SelRun(),

ErrGet()

Kategorie : Datensatzoperation

Ursache :

Die Operation wurde durch den Benutzer abgebrochen und daher nicht vollständig durchgeführt.

Dieses Resultat ist nur bei Mengenoperationen möglich, bei denen eine

Abbruchmöglichkeit für den Benutzer besteht.

Kontakt

Konstanten für Datenbankfehler

Siehe [Alle Befehle](#)

[ErrDbaAreaInUse](#)

Datenbank ist vorübergehend gesperrt.

[ErrDbaAreaLocked](#)

Datenbank ist gesperrt.

[ErrDbaAreaLockedAdmin](#)

Datenbank ist durch den Administrator gesperrt.

[ErrDbaAreaLockedDown](#)

Datenbankprozess wird beendet.

[ErrDbaAreaLockedNoStandbyOpen](#)

Standby-System steht nicht zur Verfügung.

[ErrDbaAreaLockedOpen](#)

Sperre für Login gesetzt.

[ErrDbaAreaLockedOperation](#)

Datenbank ist durch eine Serviceoperation gesperrt.

[ErrDbaAreaLockedRollback](#)

Datenbank ist gesperrt (Sperre für Rollback).

[ErrDbaAreaLockedStandby](#)

Datenbank im Standby-Modus.

[ErrDbaAreaOpen](#)

Fehler beim Öffnen der Datenbank.

[ErrDbaAreaOperationDenied](#)

Operation nicht erlaubt.

[ErrDbaAreaPassword](#)

Serverkennwort nicht korrekt.

[ErrDbaAreaRollback](#)

Datenbank ist im Rollback-Modus.

[ErrDbaAreaStandby](#)

Datenbank ist im Standby-Modus.

[ErrDbaAreaType](#)

Datenbankversion ungültig.

[ErrDbaComm](#)

Allgemeiner Kommunikationsfehler.

[ErrDbaNoArea](#)

Datenbank nicht vorhanden.

[ErrDbaNoServer](#)

Server nicht vorhanden.

[ErrDbaServerStart](#)

Fehler beim Serverstart.

[ErrDbaUserInvalid](#)

Datenbankbenutzer ist ungültig.

[ErrDbaUserLimit](#)

Maximale Benutzerzahl ist erreicht.

[ErrDbaUserSelf](#)

Ausloggen des eigenen Benutzers.

Kontakt

_ErrDbaAreaInUse

Datenbank ist vorübergehend gesperrt

Wert -806

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache : Es konnte keine genaue Ursache ermittelt werden.

Kontakt

_ErrDbaAreaLocked

Datenbank ist gesperrt

Wert -805

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist in exklusiver Benutzung.

Kontakt

_ErrDbaAreaLockedAdmin

Datenbank ist durch den Administrator gesperrt Wert -
830

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache

Die zu öffnende Datenbank ist durch den Administrator mit einer

: [Login-Sperre](#) versehen worden.

Kontakt

_ErrDbaAreaLockedDown

Datenbankprozess wird beendet

Wert -832

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache : Der Datenbankprozess wird gerade beendet. Die Datenbank kann zu einem späteren Zeitpunkt wieder geöffnet werden.

Kontakt

_ErrDbaAreaLockedNoStandbyOpen

Standby-System steht nicht zur Verfügung

Wert -836

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

In den Einstellungen der Datenbank ist eine Sperre eingetragen, für den Ursache :

Fall, dass das Standby-System nicht zur Verfügung steht (siehe

[Einstellungen der Datenbank](#)).

Kontakt

_ErrDbaAreaLockedOpen

Sperre für Login gesetzt

Wert -835

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Die zu öffnende Datenbank ist gesperrt. Die Sperre kann über die

Ursache : [Web-Administration](#) wieder freigegeben werden.

Kontakt

_ErrDbaAreaLockedOperation

Datenbank ist durch eine Serviceoperation gesperrt Wert -
831

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache :
Die zu öffnende Datenbank ist durch eine Serviceoperation (zum Beispiel
eine [Datenbankoptimierung](#)) gesperrt.

Kontakt

_ErrDbaAreaLockedRollback

Datenbank ist gesperrt (Sperre für Rollback)

Wert -834

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Die zu öffnende Datenbank ist nicht abgeschlossen, ein Rollback kann Ursache : aufgrund einer Rollback-Sperre nicht durchgeführt werden. Die

Rollback-Sperre kann über die [Web-Administration](#) aufgehoben werden.

Kontakt

_ErrDbaAreaLockedStandby

Datenbank im Standby-Modus

Wert -833

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Die zu öffnende Datenbank kann nicht geöffnet werden, da der andere

Ursache : Server des Hot-Standby-Systems die Datenbank im aktiven Betrieb hat.

Kontakt

_ErrDbaAreaOpen

Fehler beim Öffnen der Datenbank

Wert -804

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache : Die Datenbank konnte auf dem Zielserver nicht geöffnet werden.

_ErrDbaAreaOperationDenied

Operation nicht erlaubt

Wert -821

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Datenbankoperation

Ursache : Die Operation ist nicht erlaubt.

Kontakt

_ErrDbaAreaPassword

Serverkennwort nicht korrekt

Wert -808

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache :

Die zu öffnende Datenbank ist per Serverkennwort geschützt, wobei dieses nicht korrekt angegeben wurde.

Kontakt

_ErrDbaAreaRollback

Datenbank ist im Rollback-Modus

Wert -817

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Die zu öffnende Datenbank ist im Rollback-Modus und kann nicht geöffnet Ursache : werden.

Kontakt

_ErrDbaAreaStandby

Datenbank ist im Standby-Modus

Wert -827

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Die zu öffnende Datenbank ist im Standby-Modus und kann nicht geöffnet Ursache : werden.

Kontakt

_ErrDbaAreaType

Datenbankversion ungültig

Wert -807

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache : Die Version der zu öffnenden Datenbank ist nicht korrekt.

Kontakt

_ErrDbaComm

Allgemeiner Kommunikationsfehler

Wert -802

Verwandte Befehle,

Siehe [ErrGet\(\)](#), [DbaConnect\(\)](#),

[DbaConnectOpErrorCode](#)

Kategorie : Datenbankoperation

Die Verbindung zum Datenbankserver ist unterbrochen worden.

Dieser Fehler kommt beispielsweise im [SOA-Service](#) bei der Verwendung von [Datensatzbefehlen](#), wenn ein Verbindungsabbruch zu einer [verbundenen Datenbank](#) aufgetreten ist.

Ursache :



Im [Standard- und Advanced-Client](#) wird der Fehler nur zurückgegeben, wenn in der Eigenschaft [Options](#) des [Sys-Objektes](#) die Option [DbaConnectOpErrorCode](#) gesetzt ist. Ist die Option nicht gesetzt, kommt es im [Standard- und Advanced-Client](#) beim Zugriff auf die verbundene Datenbank zu einem Verbindungsabbruch.

Kontakt

_ErrDbaNoArea

Datenbank nicht vorhanden

Wert -803

Verwandte

Siehe Befehle,

DbaConnect(),

ErrGet()

Kategorie : Datenbankoperation

Ursache :
Die zu öffnende Datenbank ist auf dem Zielserver nicht eingetragen oder
nicht vorhanden.

Kontakt

[_ErrDbaNoServer](#)

Server nicht vorhanden

Wert -801

[Verwandte](#)

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Es konnte keine Verbindung mit dem Datenbankserver hergestellt Ursache : werden. Möglicherweise verhindert eine Firewall den Kontakt. Zur

Kommunikation muss auf dem Server der Port 4722 offen sein.

Kontakt

_ErrDbaServerRelease

Die Server-Version kleiner als die Client-Version Wert - 823

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie : Datenbankoperation

Ursache :

Der Fehler wird zurückgegeben, wenn die Server-Version kleiner als die Version des zu verbindenden Clients ist.

Kontakt

_ErrDbaServerStart

Fehler bei Serverstart

Wert -810

Verwandte

Siehe Befehle,

DbaConnect(),

ErrGet()

Kategorie : Datenbankoperation

Der CONZEPT 16-Server konnte den für die Datenbank eingetragenen

Ursache : Puffer nicht anlegen.

Kontakt

_ErrDbaUserInvalid

Datenbankbenutzer ist ungültig

Wert -811

Verwandte

Befehle,

Siehe DbaConnect(),

UserClear(),

ErrGet()

Kategorie : Datenbankoperation

Der angegebene Benutzer ist nicht vorhanden, das Kennwort ist nicht Ursache :
korrekt oder über den Benutzer ist kein Datenbankzugriff von einer
anderen Datenbank erlaubt.

Kontakt

_ErrDbaUserLimit

Maximale Benutzeranzahl des CONZEPT 16-Servers ist erreicht Wert **-809**

Verwandte

Siehe [Befehle](#),

[DbaConnect\(\)](#),

[ErrGet\(\)](#)

Kategorie	:	Datenbankoperation
Ursache	:	Das Benutzerlimit des Servers ist erreicht oder die Cachegröße der Zieldatenbank ist für die Benutzermenge unzureichend.

Kontakt

_ErrDbaUserSelf

Ausloggen des eigenen Benutzers

Wert -813

Verwandte

Siehe Befehle,

UserClear(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Mit dem Befehl UserClear() sollte der eigene Benutzer aus der Datenbank entfernt werden. Dies ist nicht möglich.

Kontakt

Konstanten für Verarbeitungsfehler

Siehe <u>Alle Befehle</u>	
<u>ErrAccessMode</u>	Durchführung verweigert
<u>ErrEndOfData</u>	Keine weiteren Nachrichten vorhanden
<u>ErrExists</u>	Objekt existiert bereits
<u>ErrInProgress</u>	Job-Event wird verarbeitet
<u>ErrInUse</u>	Objekt wird bereits verwendet
<u>ErrKilled</u>	Prozess wurde nicht ordnungsgemäß beendet
<u>ErrLimitExceeded</u>	Limitationen überschritten
<u>ErrLocked</u>	Objekt ist gesperrt
<u>ErrNameInvalid</u>	Ungültiger Name
<u>ErrOutOfMemory</u>	Speicher konnte nicht angefordert werden
<u>ErrRange</u>	Keine zu druckende Seite
<u>ErrSvcSessionState</u>	Zustand der Session nicht korrekt
<u>ErrSystem</u>	Nicht spezifizierbarer Fehler beim Drucken
<u>ErrType</u>	Ungültiger Typ
<u>ErrUnavailable</u>	Funktion oder Objekt steht nicht zur Verfügung
<u>ErrUnchangeable</u>	Objekt kann nicht geändert werden
<u>ErrUnerasable</u>	Objekt kann nicht gelöscht werden
<u>ErrUnknown</u>	Objekt ist unbekannt

Kontakt

_ErrAccessMode

Durchführung verweigert

Wert -57

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Die Funktion kann nicht durchgeführt werden, da sie vom System für Benutzer generell gesperrt wird.

Kontakt

_ErrEndOfData

Keine weiteren Nachrichten vorhanden

Wert -68

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Beim Lesen von Nachrichten mit MsxRead(MsxMessage, ...) sind keine weiteren Nachrichten vorhanden.

Kontakt

_ErrExists

Objekt existiert bereits

Wert -58

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Das Objekt existiert bereits mit dem gleichen Namen

Es soll ein Objekt angelegt werden, das bereits existiert. Möglicherweise wurde ein Name doppelt verwendet.

Kontakt

_ErrInProgress

Job-Event wird verarbeitet

Wert -55

Verwandte

Siehe Befehle,

ErrGet()

Dieser Fehlerwert wird bei der Anweisung JobEvent() zurückgegeben, wenn die angegebene Zeitspanne seit dem letzten Aufruf noch nicht vergangen ist.

Kontakt

_ErrLimitExceeded

Limitationen überschritten

Wert -60

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Eine oder mehrere Limitationen wurden überschritten.

Je nach verwendeter Anweisung können unterschiedliche Limitationen der Grund für diesen Fehler sein. Die Limits sind in unterschiedlichen Abschnitten erläutert:

Befehle der Benutzerverwaltung

Limitationen des Benutzersystems

Befehle für zentrale Datenobjekte

Zentrale Datenobjekte

Kontakt

_ErrInUse

Objekt wird bereits verwendet

Wert -67

Verwandte

Siehe [Befehle](#),

ErrGet()

Dieser Fehlerwert wird zurückgegeben, wenn ein Objekt in Benutzung ist. Bei [JobOpen\(\)](#) ist dies der Fall, wenn für den Job bereits ein [Kontroll-Objekt](#) angelegt wurde. Bei [UrmDelete\(\)](#) kommt der Fehlerwert, wenn ein Benutzer gelöscht werden soll, welcher noch angemeldet ist.

Kontakt

_ErrKilled

Prozess wurde nicht ordnungsgemäß beendet Wert -

66

Verwandte

Siehe Befehle,

JobErrorCode

Hat die Eigenschaft JobErrorCode den Wert _ErrKilled, wurde der Job nicht ordnungsgemäß beendet. Dies ist zum Beispiel dann der Fall, wenn der Prozess mit Hilfe des Task-Managers terminiert wurde.

Kontakt

_ErrLimitExceeded

Limitationen überschritten

Wert -60

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Eine oder mehrere Limitationen wurden überschritten.

Je nach verwendeter Anweisung können unterschiedliche Limitationen der Grund für diesen Fehler sein. Die Limits sind in unterschiedlichen Abschnitten erläutert:

Befehle der Benutzerverwaltung

Limitationen des Benutzersystems

Befehle für zentrale Datenobjekte

Zentrale Datenobjekte

Kontakt

_ErrLocked

Objekt ist gesperrt

Wert -59

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Das Objekt ist von einem anderen Benutzer gesperrt.

Kontakt

_ErrMemIVInvalid

Initialisierungsvektor ungültig

Wert -74

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Der bei MemEncrypt() oder MemDecrypt() angegebene Initialisierungsvektor ist ungültig. Dies ist beispielsweise der Fall, wenn

sich der Initialisierungsvektor nicht aus der hexadezimal bzw.

Ursache

Kontakt

_ErrMemIVLength

Initialisierungsvektor zu kurz oder zu lang

Wert -73

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Der bei MemEncrypt() oder MemDecrypt() angegebene

Ursache : Initialisierungsvektor ist zu kurz oder zu lang für den ausgewählten
Verschlüsselungsalgorithmus.

Kontakt

_ErrMemKeyInvalid

Schlüssel ungültig

Wert -69

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Der bei MemSign(), MemVerify(), MemEncrypt() oder MemDecrypt() angegebene Schlüssel ist ungültig. Dies ist beispielsweise der Fall, wenn

Ursache : **der Schlüssel im falschen Format angegeben ist oder sich nicht aus der hexadezimal bzw. Base64-kodierten Zeichenkette dekodieren lässt.**

Kontakt

_ErrMemKeyLength

Schlüssel zu kurz oder zu lang

Wert -72

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Der bei MemSign(), MemEncrypt() oder MemDecrypt() angegebene

Ursache : Schlüssel ist zu kurz für die zu signierende Nachricht oder nicht die passende Länge
für den ausgewählten Verschlüsselungsalgorithmus hat.

Kontakt

_ErrMemMsgVerify

Signatur passt nicht zur Nachricht

Wert -71

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Die bei MemVerify() angegebene Nachricht konnte nicht verifiziert

Ursache : werden, da die Signatur nicht zur Nachricht und dem öffentlichen Schlüssel passt.

Kontakt

_ErrMemSgnInvalid

Signatur ungültig

Wert -70

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Verarbeitungsfehler

Ursache : Die bei [MemVerify\(\)](#) angegebene Signatur ist ungültig.

Kontakt

_ErrNameInvalid

Ungültiger Name

Wert -56

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Das Objekt konnte nicht angelegt werden, weil der Name nicht erlaubte Zeichen beinhaltet.

Das Problem tritt auf, wenn zum Beispiel eine Benutzereigenschaft mit dem Namen '_UrmProp...' angelegt werden soll. Die Zeichenkette ist nicht erlaubt.

Kontakt

_ErrOutOfMemory

Speicher konnte nicht angefordert werden

Wert -12

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Der von dem Befehl angeforderte Speicher konnte nicht allokiert werden.

Kontakt

_ErrRange

Außerhalb des Bereichs

Wert -51

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Bei PrtJobClose() wurde keine zu druckende Seite ausgewählt, oder die

Ursache : Startseite ist größer als die Endseite.

Kontakt

_ErrSvcSessionState

Zustand der Session nicht korrekt

Wert -2920

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Eine Session kann nur dann angelegt bzw. gelöscht werden, wenn die ausgeführte Prozedur noch keine Session bzw. eine Session besitzt.

Ursache : **Ebenso kann nur dann eine Session der ausführenden Prozedur zugewiesen werden, wenn diese Prozedur noch keine Session besitzt.**

Kontakt

_ErrSystem

Nicht näher spezifizierbarer Fehler beim Drucken Wert -

53

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Bei PrtJobClose() ist ein Fehler aufgetreten, der nicht weiter spezifiziert werden kann.

Kontakt

_ErrType

Ungültiger Typ

Wert -52

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : In einem Parameter wird nicht der korrekte Typ angegeben.

Kontakt

_ErrUnavailable

Funktion oder Objekt steht nicht zur Verfügung Wert -

64

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Verarbeitungsfehler

Die Funktion kann nicht ausgeführt werden, da das Objekt oder andere

Ursache : Ressourcen, die benötigt werden, nicht zur Verfügung stehen.

Kontakt

_ErrUnchangeable

Objekt kann nicht geändert werden

Wert -62

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Es wurde versucht ein Objekt zu ändern, das nicht geändert werden darf.

Dieser Fehlerwert wird zum Beispiel zurückgegeben, wenn versucht wird den Benutzer "SUPERUSER" oder die Benutzergruppe "everyone" zu löschen.

Kontakt

_ErrUnerasable

Objekt kann nicht gelöscht werden

Wert -61

Verwandte

Siehe [Befehle](#),

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Es wurde versucht ein Objekt zu löschen, das nicht gelöscht werden darf.

Dieser Fehlerwert wird zum Beispiel zurückgegeben, wenn versucht wird den Benutzer "SUPERUSER" oder die Benutzergruppe "_everyone" zu löschen.

Kontakt

_ErrUnknown

Objekt ist unbekannt

Wert -65

Verwandte

Siehe Befehle,

ErrGet()

Kategorie: : Verarbeitungsfehler

Diese Konstante wird von den Anweisungen JobOpen() und

Ursache: : SvcSessionControl() zurückgegeben, wenn das übergebene Job-Objekt oder die übergebene Session-Id nicht vorhanden ist.

Kontakt

Konstanten für Netzwerkinformationsfehler

Siehe <u>Alle Befehle</u>	
<u>ErrNetCreate</u>	ICMP-Sockets nicht vorhanden.
<u>ErrNetIcmpID</u>	Antwortpaket mit der falschen ID.
<u>ErrNetIcmpType</u>	Antwortpaket vom falschen Typ.
<u>ErrNetNoHost</u>	Hostadresse nicht ermittelbar.
<u>ErrNetRead</u>	Fehler beim Lesen der Antwort.
<u>ErrNetReadLess</u>	Antwortpaket ungültig.
<u>ErrNetSelect</u>	Fehler bei der Socketabfrage.
<u>ErrNetWrite</u>	Anfrage nicht versendet.

Kontakt

_ErrNetCreate

ICMP-Sockets nicht vorhanden

Wert -706

NetInfo(),

Siehe ErrGet()

Kategorie : Netzwerkinformationsfehler

ICMP-Sockets nicht verfügbar. Der systemeigene Befehl "ping" arbeitet

Ursache : mit ICMP-Paketen und falls kein ICMP-Socket vorhanden ist, können die Pakete nicht verarbeitet werden.

ICMP steht für Internet Control Message Protocol.

Kontakt

_ErrNetIcmpID

Antwortpaket mit der falschen ID

Wert -720

NetInfo(),

Siehe ErrGet()

Kategorie : Netzwerkinformationsfehler

Ursache : Antwortpaket mit der falschen ID.

Kontakt

_ErrNetIcmpType

Antwortpaket vom falschen Typ

Wert -719

Siehe [NetInfo\(\)](#),

[ErrGet\(\)](#)

Kategorie : Netzwerkinformationsfehler

Ursache : Antwortpaket vom falschen Typ.

Kontakt

_ErrNetNoHost

Hostadresse nicht ermittelbar

Wert -705

NetInfo(),

Siehe ErrGet()

Kategorie : Netzwerkinformationsfehler

Ursache : Die Hostadresse eines im Netz gesuchten Rechners konnte nicht ermittelt werden.

Kontakt

_ErrNetRead

Fehler beim Lesen der Antwort

Wert -710

Siehe [NetInfo\(\)](#),

[ErrGet\(\)](#)

Kategorie : Netzwerkinformationsfehler

Ursache : Fehler beim Lesen der Antwort.

Kontakt

_ErrNetReadLess

Antwortpaket ungültig

Wert -712

NetInfo(),

Siehe ErrGet()

Kategorie : Netzwerkinformationsfehler

Ursache : Antwortpaket ungültig.

_ErrNetSelect

Fehler bei der Socketabfrage

Wert -709

Siehe [NetInfo\(\)](#),

[ErrGet\(\)](#)

Kategorie : Netzwerkinformationsfehler

Ursache : Fehler bei der Socketabfrage.

Kontakt

_ErrNetWrite

Anfrage nicht versendet

Wert -713

Siehe [NetInfo\(\)](#),
[ErrGet\(\)](#)

Kategorie : Netzwerkinformationsfehler

Ursache : Anfrage nicht versendet.

Kontakt

Konstanten für Socketfehler

Alle

Siehe Befehle

[ErrSckBindFailed](#)

Es konnte keine Verbindung zum Zielport aufgebaut werden.

[ErrSckConnectFailed](#)

Es konnte keine Verbindung zum Zielhost aufgebaut werden.

[ErrSckCreate](#)

Der Socket konnte nicht angelegt werden.

[ErrSckDown](#)

Der Socket wurde außerhalb der Applikation geschlossen.

[ErrSckHostUnknown](#)

Der Name des Zielhosts konnte nicht in eine IP-Adresse übersetzt werden.

[ErrSckNoLib](#)

Das TCP/IP-Protokoll konnte nicht initialisiert werden.

[ErrSckProxyAuthFailed](#)

Authentifikation beim Proxy fehlgeschlagen.

[ErrSckProxyConnectFailed](#) Es

konnte keine Verbindung zum Proxy aufgebaut

werden.

[ErrSckProxyFailed](#)

Kommunikation mit dem Proxy ist fehlgeschlagen.

[ErrSckProxyRead](#)

Bei der Kommunikation mit dem Proxy ist ein Lesefehler aufgetreten.

[ErrSckProxyRefused](#)

Der Proxy hat den Verbindungswunsch abgelehnt.

[ErrSckProxyUnknown](#)

Der Name des Proxys konnte nicht in eine IP-Adresse übersetzt werden.

[ErrSckProxyWrite](#)

Bei der Kommunikation mit dem Proxy ist ein

[ErrSckRead](#)

Schreibfehler aufgetreten.

[ErrSckReadOverflow](#)

Beim Lesen des Sockets ist ein Fehler aufgetreten

(beispielsweise durch Abbruch der Verbindung).

Beim Lesen des Sockets mit der Option [_SckLine](#) ist die empfangene Zeile länger als die angegebene Variable.

Bei der Socketabfrage ist ein Fehler aufgetreten

(beispielsweise durch Abbruch der Verbindung).

Fehler bei der Überprüfung des serverseitigen

Zertifikats.

Fehler beim Aufbau einer verschlüsselten Verbindung.

Beim Lesen des Sockets ist ein Fehler aufgetreten

(beispielsweise durch Abbruch der Verbindung).

[ErrSckTlsCertificateVerify](#)

[ErrSckTlsConnect](#)

[ErrSckWrite](#)

Kontakt

_ErrSckBindFailed

Es konnte keine Zuordnung zum angegebenen Port erfolgen Wert -714

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Möglicherweise ist der angegeben Port bereits in Benutzung.

Kontakt

_ErrSckConnectFailed

Es konnte keine Verbindung zum Zielhost aufgebaut werden Wert -707

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Es konnte keine Verbindung zum Zielhost aufgebaut werden.

Die Fehlernummer wird ebenfalls in den [Log-Einträgen](#) des Servers ausgegeben, wenn im Hot-Standby Betrieb einer der beteiligten Server seinen Partner-Server nicht findet.

Kontakt

_ErrSckCreate

Der Socket konnte nicht angelegt werden

Wert -706

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Der Socket konnte nicht angelegt werden.

Kontakt

_ErrSckDown

Der Socket wurde von der anderen Seite geschlossen Wert -717

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Der Socket wurde außerhalb der Applikation geschlossen.

Kontakt

[_ErrSckHostUnknown](#)

Der Name des Zielhosts konnte nicht in eine IP-Adresse übersetzt werden Wert -705

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Der Name des Zielhosts konnte nicht in eine IP-Adresse übersetzt werden. Die Fehlernummer wird ebenfalls in den [Log-Einträgen](#) des Servers ausgegeben, wenn im Hot-Standby Betrieb einer der beteiligten Server den Namen seines Partner-Servers nicht in eine IP-Adresse auflösen kann.

Kontakt

[_ErrSckNoLib](#)

Das TCP/IP-Protokoll konnte nicht initialisiert werden Wert - 701

[Verwandte](#)

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Das TCP/IP-Protokoll konnte nicht initialisiert werden.

_ErrSckProxyAuthFailed

Authentifikation beim Proxy fehlgeschlagen Wert -

733

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Der angegebene Benutzer oder das angegebene Passwort ist ungültig.

Kontakt

_ErrSckProxyConnectFailed

Es konnte keine Verbindung zum Proxy aufgebaut werden Wert -
725

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Es konnte keine Verbindung zum Proxy aufgebaut werden.

Kontakt

_ErrSckProxyFailed

Kommunikation mit dem Proxy ist fehlgeschlagen Wert - 732

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Socket-Funktionen

Vom Proxy wurde eine ungültige Antwort geschickt oder die angeforderte

Ursache : Authentifizierungsmethode wird von Proxy nicht unterstützt.

Kontakt

_ErrSckProxyRead

Bei der Kommunikation mit dem Proxy ist ein Lesefehler aufgetreten Wert -726

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Socket-Funktionen

Ursache : Bei der Kommunikation mit dem Proxy ist ein Lesefehler aufgetreten.

Kontakt

_ErrSckProxyRefused

Der Proxy hat den Verbindungswunsch abgelehnt.

Wert -724

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Socket-Funktionen

Der Proxy hat den Verbindungswunsch abgelehnt. Dies kann auf Grund

**Ursache : von technischen Problemen (keine Netzwerkverbindung) oder auf Grund von
Vorgaben beim Proxy erfolgen.**

Kontakt

_ErrSckProxyUnknown

Der Name des Proxys konnte nicht in eine IP-Adresse übersetzt werden Wert -722

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Der Name des Proxys konnte nicht in eine IP-Adresse übersetzt werden.

Kontakt

_ErrSckProxyWrite

Bei der Kommunikation mit dem Proxy ist ein Schreibfehler aufgetreten Wert -727

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Bei der Kommunikation mit dem Proxy ist ein Schreibfehler aufgetreten.

Kontakt

_ErrSckRead

Beim Lesen des Sockets ist ein Fehler aufgetreten Wert -
710

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

**Ursache : Beim Lesen des Sockets ist ein Fehler aufgetreten
(beispielsweise durch Abbruch der Verbindung).**

Die Fehlernummer wird ebenfalls in den [Log-Dateien](#) des Servers ausgegeben, wenn beim Lesevorgang auf die Hot-Standby Direktverbindung ein Problem auftritt.

Kontakt

[_ErrSckReadOverflow](#)

empfangene Zeile länger als die angegebene Variable Wert -711

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Ursache : Beim Lesen des Sockets mit der Option [SckLine](#) ist die empfangene Zeile länger als die angegebene Variable.

Kontakt

_ErrSckSelect

Bei der Socketabfrage ist ein Fehler aufgetreten Wert -
709

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Socket-Funktionen

Ursache : Bei der Socketabfrage ist ein Fehler aufgetreten
(beispielsweise durch Abbruch der Verbindung).

Die Fehlernummer wird ebenfalls in den Log-Dateien des Servers ausgegeben, wenn beim Lese- oder Schreibvorgang auf die Hot-Standby Direktverbindung ein Problem auftritt.

Kontakt

[_ErrSckTlsCertificateVerify](#)

Fehler bei der Überprüfung des serverseitigen Zertifikats Wert -745

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

Es ist ein Fehler bei der Überprüfung des serverseitigen Zertifikats Ursache : aufgetreten. Der Fehler kann über die Eigenschaft [CertificateError](#) des [Sys-Objekts](#) ermittelt werden.

Kontakt

_ErrSckTlsConnect

Fehler beim Aufbau einer verschlüsselten Verbindung Wert -

741

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Socket-Funktionen

Der angegebene Server unterstützt keines der angegebenen verschlüsselten Verbindungsprotokolle oder der TLS-Handshake ist fehlgeschlagen. Eine Mögliche Ursache könnte sein, dass der Server

Ursache : **mehrere Namen hat und der korrekte Server per "Server Name Indication" (SckTlsSNI) beim Verbindungsaufbau ausgewählt werden muss.**

Kontakt

_ErrSckWrite

Beim Schreiben des Sockets ist ein Fehler aufgetreten Wert **-713**

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Socket-Funktionen

**Ursache : Beim Schreiben des Sockets ist ein Fehler aufgetreten
(beispielsweise** durch Abbruch der Verbindung).

Die Fehlernummer wird ebenfalls in den [Log-Dateien](#) des Servers ausgegeben, wenn beim Schreiben auf die Hot-Standby Direktverbindung ein Problem auftritt.

Kontakt

Konstanten für Fehler bei der Mehrfachselektion Siehe

Alle **Befehle**

ErrMsdExists Objekt ist bereits selektiert.

ErrMsdNotFound Objekt ist nicht selektiert.

Kontakt

_ErrMsdExists

Objekt ist bereits selektiert

Wert -1750

Siehe [WinMsdInsert\(\)](#)

Dieser Wert wird von der Anweisung [WinMsdInsert\(\)](#) zurückgegeben, wenn das übergebene Element bereits in der selektierten Menge vorhanden ist.

Kontakt

_ErrMsdNotFound

Objekt nicht selektiert

Wert -1751

Siehe WinMsdDelete()

Dieser Wert wird von der Anweisung WinMsdDelete() zurückgegeben, wenn das übergebene Element nicht in der selektierten Menge vorhanden ist.

Kontakt

Konstanten für Fehler bei dynamischen Selektionen

Siehe <u>Alle Befehle</u>	
<u>ErrParserEndOfText</u>	Abfrage unvollständig.
<u>ErrParserIllegalElement</u>	Ungültiges Element.
<u>ErrParserInvalidChar</u>	Unerlaubtes Zeichen.
<u>ErrParserInvalidConst</u>	Ungültige Konstante.
<u>ErrParserMissingComma</u>	Komma fehlt.
<u>ErrParserMissingParenthesis</u>	Klammer fehlt.
<u>ErrParserOutOfRange</u>	Unzulässiger Wert.
<u>ErrParserStringOverflow</u>	Zeichenkette zu lang.
<u>ErrParserSyntax</u>	Syntaxfehler in der Abfrage.
<u>ErrParserUnknownID</u>	Unbekannter Anker.
<u>ErrParserWrongType</u>	Falscher Typ angegeben.
<u>ErrSelCodeOverflow</u>	Interner Fehler bei der Übersetzung und Speicherung einer Selektion.
<u>ErrSelIllegalOperator</u>	Unzulässiger Operator.
<u>ErrSelInvalidField</u>	Ungültiges Feld.
<u>ErrSelInvalidKey</u>	Vorauswahl mit falschem Schlüssel.
<u>ErrSelNoQuery</u>	Verwendung einer undefinierten Abfrage.
<u>ErrSelQueryOverflow</u>	Abfrage zu lang.
<u>ErrSelResultSet</u>	Fehler bei einer Ergebnismenge.
<u>ErrSelTableOverflow</u>	Interner Fehler bei der Übersetzung und Speicherung einer Selektion.
<u>ErrSelUnknownField</u>	Unbekanntes Feld.
<u>ErrSelUnknownOrInvalidLink</u>	Unbekannte oder ungültige Verknüpfung.

Kontakt

_ErrParserEndOfText

Abfrage unvollständig

Wert -2100

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : Die Abfrage ist nicht vollständig definiert.

_ErrParserIllegalElement

Ungültiges Element

Wert -2108

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Ursache : Ein in der Abfrage enthaltenes Element ist nicht gültig.

Kontakt

_ErrParserInvalidChar

Unerlaubtes Zeichen

Wert -2101

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Ursache : In der Abfrage ist ein unerlaubtes Zeichen enthalten.

Kontakt

_ErrParserInvalidConst

Ungültige Konstante

Wert -2102

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : In der Abfrage ist eine ungültige Konstante enthalten.

_ErrParserMissingComma

Komma fehlt

Wert -2110

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Ursache : In der Abfrage fehlt ein Komma.

_ErrParserMissingParenthesis

Klammer fehlt

Wert -2109

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Ursache : In der Abfrage fehlt eine oder mehrere Klammern.

Kontakt

_ErrParserOutOfRange

Unzulässiger Wert

Wert -2104

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : In der Abfrage ist ein Wert außerhalb seines Definitionsbereichs angegeben.

Kontakt

_ErrParserStringOverflow

Zeichenkette zu lang

Wert -2105

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : Eine Zeichenkette innerhalb der Abfrage ist zu lang.

_ErrParserSyntax

Syntaxfehler

Wert -2107

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Ursache : In der Abfrage befindet sich ein Syntaxfehler.

Kontakt

_ErrParserUnknownID

Unbekannter Anker

Wert -2106

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

**Ursache : In der Abfrage wurde ein Anker angegeben, der zuvor nicht mit
SelAddLink() angegeben wurde.**

Kontakt

_ErrParserWrongType

Falscher Typ angegeben

Wert -2103

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : In der Abfrage wurde ein falscher Typ verwendet.

Kontakt

_ErrSelCodeOverflow

Interner Fehler bei der Übersetzung und Speicherung einer Selektion Wert -2208

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Beim Übersetzen und Speichern einer dynamischen Selektion mit der

Ursache : Anweisung [SelStore\(\)](#) ist ein Code länger als 128 KB entstanden.

Kontakt

_ErrSelIllegalOperator

Ungültiger Operator

Wert -2204

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : In der Abfrage wurde ein ungültiger Operator (zum Beispiel ><) angegeben.

Kontakt

_ErrSelInvalidField

Feld ungültig

Wert -2202

Verwandte

Siehe [Befehle](#),

[SelDefQuery\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : In der Abfrage ist ein ungültiges Feld angegeben.

Kontakt

_ErrSelInvalidKey

Vorauswahl mit falschem Schlüssel

Wert -2210

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Bei der Anweisung SelDefQuery() ist für eine Vorauswahl ein Schlüssel angegeben, der nicht verwendet werden kann. Mindestens ein

Ursache : Schlüsselfeld muss im Abfragekriterium vorhanden sein. Der Fehler tritt auch dann auf, wenn der angegebene Schlüssel nicht existiert oder eines der Schlüsselfelder SOUNDEX 1 oder SOUNDEX 2 aktiviert hat.

Kontakt

_ErrSelNoQuery

Verwendung einer undefinierten Abfrage

Wert -2209

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : Innerhalb der dynamischen Selektion wird mit [SelAddLink\(\)](#) auf eine nicht vorhandene Abfrage verwiesen.

Kontakt

_ErrSelQueryOverflow

Abfrage zu lang

Wert -2205

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

**Ursache : Die Zeichenkette der Abfrage ist zu lang oder es wurden mehr als 100
Abfrageelemente angegeben.**

Kontakt

_ErrSelResultSet

Fehler bei einer Ergebnismenge

Wert -2206

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Selektionsdefinition

Mit der Anweisung SelStore() soll eine Selektion gespeichert werden, die keine Ergebnismenge besitzt. Die Ergebnismenge muss bei SelCreate()

Ursache : oder SelAddResult() angegeben werden. Der Fehler tritt auch auf, wenn zu einer definierten Ergebnismenge keine Abfrage vorhanden ist oder notwendige Ergebnismengen nicht definiert sind.

Kontakt

_ErrSelTableOverflow

Interner Fehler bei der Übersetzung und Speicherung einer Selektion Wert -2207

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Selektionsdefinition

Beim Übersetzen und Speichern einer dynamischen Selektion mit der Ursache : Anweisung SelStore() ist es zum Überlauf interner Tabellen gekommen.

Kontakt

_ErrSelUnknownField

Feldname unbekannt

Wert -2201

Verwandte

Siehe Befehle,

SelDefQuery(),

ErrGet()

Kategorie : Selektionsdefinition

Ursache : Es wurde ein unbekanntes Feld angegeben.

Kontakt

_ErrSelUnknownOrInvalidLink

Verknüpfung unbekannt oder ungültig

Wert -2203

Verwandte

Befehle,

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

[SelAddResult\(\)](#),

[ErrGet\(\)](#)

Kategorie : Selektionsdefinition

Ursache : Der angegebene Anker entspricht keiner Verknüpfung bzw. keiner gültigen Verknüpfung.

Kontakt

Konstanten für Fehler bei Storage-Objekten

Alle

Siehe Befehle

ErrStoInvalidFormat Ungültiges Format.

ErrStoLocked Das Storage-Objekt ist gesperrt.

ErrStoNameInvalid Der Name des Storage-Objekts oder -Verzeichnisses ist ungültig.

ErrStoNoData Keine Daten vorhanden.

ErrStoNoFile Das Storage-Objekt existiert nicht.

ErrStoNoPath Das Storage-Verzeichnis existiert nicht.

ErrStoOperation Operation kann nicht ausgeführt werden.

Kontakt

_ErrStoInvalidFormat

Ungültiges Format

Wert -1511

Verwandte

Befehle,

Siehe [ErrGet\(\)](#),

unterstützte

Dateiformate

Kategorie : Storage-Befehle

Ursache : Es wurde versucht ein ungültiges Format zu importieren.

Kontakt

_ErrStoLocked

Das Storage-Objekt ist gesperrt

Wert -1506

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Storage-Befehle

Ursache : Das Storage-Objekt ist von einem anderen Benutzer gesperrt. Die Operation kann aber nur dann durchgeführt werden, wenn das Objekt nicht oder vom eigenen Benutzer gesperrt ist.

Kontakt

_ErrStoNameInvalid

Name des Storage-Objekts/-Verzeichnisses ist ungültig Wert -
1501

Siehe Befehle,
ErrGet()

Kategorie : Storage-Befehle

Ursache : Es ist ein ungültiger Name angegeben worden. Der Name darf nur aus den Buchstaben A...z den Ziffern 0...9 sowie dem "_" und dem ":" bestehen, wobei das erste Zeichen ein Buchstabe sein muss. Er darf maximal 40 Zeichen lang sein.

Kontakt

_ErrStoNoData

Keine Daten vorhanden

Wert -1504

[Verwandte](#)

[Befehle](#),

Siehe [StoImport\(\)](#),

[StoExport\(\)](#),

[ErrGet\(\)](#)

Kategorie : Storage-Befehle

Ursache : Die angegebene externe Datei oder das angegebene Objekt beinhaltet keine Daten.

Kontakt

_ErrStoNoFile

Storage-Objekt existiert nicht

Wert -1503

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Storage-Befehle

Ursache : Das angegebene Storage-Objekt existiert nicht.

Kontakt

_ErrStoNoPath

Storage-Verzeichnis existiert nicht.

Wert -1502

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Storage-Befehle

Ursache : Das angegebene Storage-Verzeichnis existiert nicht.

Kontakt

_ErrStoOperation

Operation kann nicht ausgeführt werden

Wert -1510

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Storage-Befehle

Ursache : Es wurde versucht ein Storage-Objekt im Wurzelverzeichnis anzulegen.

Es wurde versucht ein Tile in ein falsches Verzeichnis zu importieren.

Es wurde versucht ein Oberflächen-Objekt zu importieren.

Es wurde versucht die Optionen _StoCreate und _StoDirectory zu kombinieren.

Kontakt

Konstanten für Fehler der Tapi-Schnittstelle

Siehe [Alle Befehle](#)

<u>ErrTapiBadAddr</u>	Kein Anschluss
<u>ErrTapiBusy</u>	Besetzt
<u>ErrTapiCallID</u>	Ungültige Call-ID
<u>ErrTapiCallState</u>	Operation in diesem Status nicht zulässig
<u>ErrTapiDevName</u>	Ungültiges TAPI-Device
<u>ErrTapiDialString</u>	Keine wählbare Nummer
<u>ErrTapiDialTimeout</u>	TAPI-Timeout überschritten
<u>ErrTapiFailed</u>	Operation nicht erfolgreich
<u>ErrTapiInstall</u>	TAPI-Schnittstelle nicht korrekt installiert
<u>ErrTapiInUse</u>	TAPI-Gerätetreiber in Benutzung
<u>ErrTapiMediaMode</u>	Nicht unterstützter Medienmodus
<u>ErrTapiMemory</u>	Kein Speicher
<u>ErrTapiNoConnect</u>	Nicht verbunden
<u>ErrTapiNoListen</u>	TapiDevice wird nicht überwacht
<u>ErrTapiNotOwner</u>	Keine Besitzrechte
<u>ErrTapiReinit</u>	TAPI-Konfiguration wurde verändert
<u>ErrTapiReject</u>	Anruf abgelehnt
<u>ErrTapiUnavail</u>	Operation oder Verbindung nicht verfügbar
<u>ErrTapiUnknown</u>	Unbekannter Fehler
<u>ErrTapiVersion</u>	Falsche TAPI-Version

Kontakt

_ErrTapiBadAddr

Kein Anschluss

Wert -1814

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Kein Anschluss, die Verbindung wurde zurückgewiesen.

Kontakt

_ErrTapiBusy

Besetzt

Wert -1813

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Die angerufene Gegenstelle ist besetzt.

Kontakt

_ErrTapiCallID

Ungültige Call-ID

Wert -1819

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Dem Befehl wurde eine ungültige Call-ID übergeben.

Kontakt

_ErrTapiCallState

Operation in diesem Status unzulässig

Wert -1817

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Es wurde eine Operation ausgeführt, die in dem Status des Anrufs nicht zulässig ist.

_ErrTapiDevName

Ungültiges TAPI-Device

Wert -1803

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Tapi-Fehler

Ursache : Name des Devices existiert nicht.

Kontakt

_ErrTapiDialString

Keine wählbare Nummer

Wert -1805

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Es wurde keine Telefonnummer angegeben.

Der Fehler tritt auf, wenn in einem Wählstring illegale Zeichen (zum Beispiel Buchstaben) enthalten sind.

Kontakt

_ErrTapiDialTimeout
TAPI-Timeout überschritten
Wert -1806

Verwandte

Siehe **Befehle**,
ErrGet()

Kategorie : Tapi-Fehler

Ursache : Das TAPI-Device meldet sich nicht.

Das TAPI-Device wurde angesprochen und es hat sich nicht innerhalb der nächsten 60 Sekunden gemeldet.

Kontakt

_ErrTapiFailed

Operation nicht erfolgreich

Wert -1810

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Tapi-Fehler

Ursache : Operation ist aus nicht näher spezifizierten Gründen fehlgeschlagen.

Kontakt

_ErrTapiInstall

TAPI-Schnittstelle nicht korrekt installiert

Wert -1807

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Die TAPI-Schnittstelle ist nicht korrekt installiert worden.

Kontakt

_ErrTapiInUse

TAPI-Gerätetreiber in Benutzung

Wert -1804

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Der TAPI-Gerätetreiber wird von einem anderen Programm benutzt.

Kontakt

_ErrTapiMediaMode

Nicht unterstützter Medienmodus

Wert -1812

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Der beim Modem eingestellte Medienmodus wird nicht unterstützt.

Kontakt

_ErrTapiMemory

Kein Speicher

Wert -1809

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Es konnte nicht genug Hauptspeicher allokiert werden.

Kontakt

_ErrTapiNoConnect

Nicht verbunden

Wert -1815

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Die Gegenstelle hebt nicht ab, die Gegenstelle lehnt eine Verbindung ab oder es gibt kein Freizeichen.

Kontakt

_ErrTapiNoListen

TapiDevice wird nicht überwacht

Wert -1821

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Dem Befehl wurde ein TapiDevice übergeben, das nicht mit TapiListen() überwacht wird.

Kontakt

_ErrTapiNotOwner

Keine Besitzrechte

Wert -1820

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Es wurde versucht eine TAPI-Funktion durchzuführen, die Besitzrechte an dem TAPI-Gerät voraussetzt. Diese Besitzrechte sind nicht vorhanden.

Kontakt

_ErrTapiReinit

TAPI-Konfiguration wurde verändert

Wert -1808

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Tapi-Fehler

Ursache : Die Konfiguration der TAPI-Schnittstelle wurde verändert.

Kontakt

_ErrTapiReject

Anruf abgelehnt

Wert -1816

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Tapi-Fehler

Ursache : Der entfernte Gesprächspartner hat den Anruf abgelehnt.

Kontakt

_ErrTapiUnavail

**Operation oder Verbindung nicht verfügbar Wert -
1811**

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Tapi-Fehler

Ursache : Es sollte eine Operation durchgeführt werden, die nicht von der TAPI-Schnittstelle unterstützt wird.

Kontakt

_ErrTapiUnknown

Unbekannter Fehler

Wert -1801

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Tapi-Fehler

Ursache : Unbekannter oder interner Fehler.

Die TAPI-Schnittstelle liefert einen unbekannten oder internen Fehler zurück.

Kontakt

_ErrTapiVersion

Falsche TAPI-Version

Wert -1802

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Tapi-Fehler

**Ursache : Die TAPI-Schnittstelle oder das TAPI-Device hat eine Versionsnummer,
die nicht unterstützt wird.**

**Die TAPI-Schnittstelle muss mindestens die Version 2.0 und der Gerätetreiber die Version 1.4
besitzen, um von CONZEPT 16 angesprochen zu werden.**

**Die TAPI-Version des Gerätetreibers kann über die Systemeigenschaft Version des Gerätetreibers
ermittelt werden.**

Kontakt

Konstanten für Dateibearbeitungsfehler (extern)

Siehe [Alle Befehle](#)

<u>ErrFsiAccessDenied</u>	Zugriff verweigert
<u>ErrFsiCurrentDir</u>	Aktuelles Verzeichnis kann nicht gelöscht werden
<u>ErrFsiDriveInvalid</u>	Ungültige Laufwerksangabe
<u>ErrFsiExists</u>	Datei existiert bereits
<u>ErrFsiHdlInvalid</u>	Ungültiger Dateideskriptor
<u>ErrFsiInvalidFormat</u>	Ungültiges Dateiformat
<u>ErrFsiLockViolation</u>	Sperrkonflikt in externer Datei
<u>ErrFsiNoFile</u>	Externe Datei nicht vorhanden
<u>ErrFsiNoPath</u>	Pfad nicht vorhanden
<u>ErrFsiOpenFailed</u>	Externe Datei konnte nicht geöffnet oder angelegt werden
<u>ErrFsiOpenOverflow</u>	Zuviele offene Dateien
<u>ErrFsiOther</u>	Unbekannter Fehler beim Arbeiten mit externen Dateien
<u>ErrFsiReadFault</u>	Fehler beim Lesen einer externen Datei
<u>ErrFsiSharingViolation</u>	Zugriffskonflikt bei externer Datei
<u>ErrFsiWriteFault</u>	Fehler beim Schreiben einer externen Datei

Kontakt

_ErrFsiAccessDenied

Zugriff verweigert

Wert -23

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Externe Dateioperation

Ursache : Auf die externe Datei oder das Verzeichnis kann aufgrund unzureichender Berechtigung nicht zugegriffen werden.

Kontakt

_ErrFsiCurrentDir

Aktuelles Verzeichnis kann nicht gelöscht werden Wert -26

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Externe Dateioperation

Ursache : Es wurde versucht, das aktuelle Verzeichnis zu löschen.

Kontakt

_ErrFsiDriveInvalid

Ungültige Laufwerksangabe

Wert -25

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : **Externe Dateioperation**

Ursache : **Auf das angegebene Laufwerk kann nicht zugegriffen werden.**

Kontakt

_ErrFsiExists

Datei existiert bereits

Wert -40

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#),

[FsiCreateNew](#)

Kategorie : Externe Dateioperation

Ursache : Die externe Datei kann nicht angelegt werden, da sie bereits existiert.

Kontakt

_ErrFsiHdlInvalid

Ungültiger Dateideskriptor

Wert -24

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Der benutzte Deskriptor des Betriebssystems ist ungültig.

Kontakt

_ErrFsiInvalidFormat

Ungültiges Dateiformat

Wert -35

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Externe Dateioperation

Ursache : Das Format der externen Datei kann nicht verwendet werden.

Kontakt

_ErrFsiLockViolation

Sperrkonflikt in externer Datei

Wert -28

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Bei einer Sperr-, Lese- oder Schreiboperation ist ein Konflikt mit existierenden Sperren anderer Benutzer aufgetreten.

Kontakt

[_ErrFsiNoFile](#)

Externe Datei oder Verzeichnis nicht vorhanden Wert -

20

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Externe Dateioperation

Ursache : Die angegebene Datei oder das angegebene Verzeichnis wurde nicht gefunden.

Bei dem Befehl [FsiPathChange\(\)](#) wird dieser Fehler zurückgegeben, wenn das zuletzt angegebene Verzeichnis nicht vorhanden ist. Existiert der Pfad zu diesem Verzeichnis nicht, wird [_ErrFsiNoPath](#) zurückgegeben.

Kontakt

_ErrFsiNoPath

Pfad nicht vorhanden

Wert -21

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Der angegebene Pfad wurde nicht gefunden.

Der Pfad zu der angegebenen Datei oder dem Verzeichnis existiert nicht. Diese Fehlermeldung trifft keine Aussage über die Existenz der Datei oder des Verzeichnisses.

Kontakt

_ErrFsiOpenFailed

Externe Datei konnte nicht geöffnet oder angelegt werden Wert -29

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Externe Dateioperation

Ursache : Datei existiert nicht.

Berechtigung unzureichend.

Dateiname ungültig.

Kontakt

_ErrFsiOpenOverflow

Zuviele offene Dateien

Wert -22

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Externe Dateioperation

Ursache : Die maximale Anzahl von offenen Dateien ist erreicht.

Kontakt

_ErrFsiOther

Unbekannter Fehler beim Arbeiten mit externen Dateien Wert -39

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Es trat ein Fehler beim Arbeiten mit externen Dateien auf, der nicht durch andere Fehlerkonstanten abgedeckt ist

Die Fehlermeldung wird ebenfalls zurückgegeben, wenn ein ungültiger Dateiname (zum Beispiel mit illegalen Zeichen) angegeben wurde.

Kontakt

_ErrFsiReadFault

Fehler beim Lesen einer externen Datei

Wert -31

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Die externe Datei konnte nicht ordnungsgemäß gelesen werden

Kontakt

_ErrFsiSharingViolation

Zugriffskonflikt bei externer Datei

Wert -27

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Die externe Datei kann aufgrund der aktuellen Zugriffssituation nicht geöffnet werden.
Dieser Fall tritt beispielsweise ein, wenn ein Anwender die Datei im exklusiven Zugriff hat, und ein weiterer Benutzer diese Datei öffnen möchte.

Kontakt

_ErrFsiWriteFault

Fehler beim Schreiben einer externen Datei Wert -

32

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Externe Dateioperation

Ursache : Die externe Datei konnte nicht ordnungsgemäß geschrieben werden

Kontakt

Konstanten für Fehler des OEM-Kits

Siehe [Alle Befehle](#)

<u>ErrOemDbaLock</u>	Datenbank ist im Mehrbenutzerbetrieb
<u>ErrOemInvalidFormat</u>	Definitionsdatei im falschen Format
<u>ErrOemOpenDesigner</u>	Prozedur wurde aus dem Designer gestartet
<u>ErrOemOpenFailed</u>	Öffnen der Definitionsdatei fehlgeschlagen
<u>ErrOemOpenFrame</u>	Ein oder mehrere Dialoge sind noch offen
<u>ErrOemOutOfSpace</u>	Speicherplatz unzureichend
<u>ErrOemPassword</u>	Kennwort ungültig oder OEM-Kit nicht vorhanden

Kontakt

_ErrOemDbaLock

Datenbank ist im Mehrbenutzerbetrieb

Wert -1502

Verwandte

Siehe Befehle,

OEMLoad(),

ErrGet()

Kategorie : OEM-Kit

Ursache : Datenbank ist im Mehrbenutzerbetrieb.

Kontakt

_ErrOemInvalidFormat

Definitionsdatei im falschen Format

Wert -1504

Verwandte

Siehe Befehle,

OEMLoad(),

ErrGet()

Kategorie : OEM-Kit

Ursache : Definitionsdatei hat nicht das korrekte Format oder wurde von einer anderen CONZEPT 16-Clientversion erstellt.

Kontakt

_ErrOemOpenDesigner

Prozedur wurde aus dem Designer gestartet Wert -

1506

Verwandte

Siehe **Befehle**,

OEMLoad(),

ErrGet()

Kategorie : OEM-Kit

Ursache : Die Prozedur wurde aus dem Designer gestartet.

Kontakt

_ErrOemOpenFailed

Öffnen der Definitionsdatei fehlgeschlagen

Wert -1503

Verwandte

Befehle,

Siehe OEMLoad(),

OEMSav(),

ErrGet()

Kategorie : OEM-Kit

Ursache : Öffnen der Definitionsdatei fehlgeschlagen.

Kontakt

_ErrOemOpenFrame

Ein oder mehrere Dialoge sind noch offen

Wert -1507

Verwandte

Siehe [Befehle](#),

[OEMLoad\(\)](#),

[ErrGet\(\)](#)

Kategorie : OEM-Kit

Ursache : Ein oder mehrere Dialoge sind noch offen.

Kontakt

_ErrOemOutOfSpace

Speicherplatz unzureichend

Wert -1505

Verwandte

Siehe [Befehle](#),

[OEMLoad\(\)](#),

[ErrGet\(\)](#)

Kategorie : OEM-Kit

Ursache : Der Speicherplatz ist unzureichend.

Kontakt

_ErrOemPassword

**Kennwort ungültig oder OEM-Kit nicht vorhanden Wert -
1501**

Verwandte

Befehle,

Siehe OEMLoad(),

OEMSav(),

ErrGet()

Kategorie : OEM-Kit

Ursache : OEM-Kit ist nicht vorhanden.

Kennwort des OEM-Kits ist ungültig.

Kontakt

Konstanten für Fehler der binären Objekte

Siehe	<u>Alle Befehle</u>
<u>ErrBinData</u>	Falsches Codewort angegeben
<u>ErrBinDecryption</u>	Falsches Codewort angegeben
<u>ErrBinDirNotEmpty</u>	Verzeichnis ist nicht leer
<u>ErrBinExists</u>	Binäres Objekt existiert bereits
<u>ErrBinLocked</u>	Das binäre Objekt ist gesperrt
<u>ErrBinNameInvalid</u>	Ungültiger Name
<u>ErrBinNoData</u>	Keine Daten vorhanden
<u>ErrBinNoFile</u>	Binäres Objekt nicht vorhanden
<u>ErrBinNoLock</u>	Binäres Objekt nicht gesperrt
<u>ErrBinNoPath</u>	Pfad nicht vorhanden
<u>ErrBinOperation</u>	Operation kann nicht ausgeführt werden

Kontakt

[_ErrBinData](#)

Falsches Codewort angegeben oder allgemeiner Fehler aufgetreten Wert **-1509**

[Verwandte Befehle](#),

Siehe [ErrGet\(\)](#),

[ErrBinDecryption](#)

Kategorie : Binäre Objekte

Ursache : Es ist kein oder der falsche Verschlüsselungscode angegeben worden oder ein allgemeiner Fehler aufgetreten.

Kontakt

_ErrBinDecryption

Falsches Codewort angegeben

Wert -1512

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#),

[ErrBinData](#)

Kategorie : Binäre Objekte

Ursache : Es ist kein oder der falsche Verschlüsselungscode angegeben worden.

Dieser Fehler tritt nur auf, wenn bei [BinExport\(\)](#) oder [BinReadMem\(\)](#) die Option [BinErrorDecryption](#) angegeben wurde.

Kontakt

_ErrBinDirNotEmpty

Verzeichnis ist nicht leer

Wert -1508

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Binäre Objekte

Ursache : Die verwendete Operation kann nur bei einem leeren Verzeichnis durchgeführt werden.

Kontakt

_ErrBinExists

Binäres Objekt existiert bereits

Wert -1507

Verwandte

Siehe [Befehle](#),

[BinOpen\(\)](#),

[ErrGet\(\)](#)

Kategorie : Binäre Objekte

Ursache : Es existiert bereits ein binäres Objekt mit dem gleichen Namen.

Kontakt

_ErrBinLocked

Das binäre Objekt ist gesperrt

Wert -1506

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Binäre Objekte

Ursache : Das binäre Objekt ist von einem anderen Benutzer gesperrt. Die Operation kann aber nur dann durchgeführt werden, wenn das Objekt nicht oder vom eigenen Benutzer gesperrt ist.

Kontakt

_ErrBinNameInvalid

Ungültiger Name

Wert -1501

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Binäre Objekte

Ursache : Der angegebene Name ist ungültig. In Namen von binären Objekten dürfen keine Steuerzeichen oder die Zeichen "*" und "?" vorkommen. Beim Anlegen eines binären Verzeichnisses, darf der Name nicht auf "\\" enden.

Kontakt

_ErrBinNoData

Keine Daten vorhanden

Wert -1504

Verwandte

Befehle,

Siehe BinImport(),

BinExport(),

ErrGet()

Kategorie : Binäre Objekte

Ursache : Die angegebene externe Datei oder das angegebene Objekt beinhaltet keine Daten.

Kontakt

_ErrBinNoFile

Binäres Objekt nicht vorhanden

Wert -1503

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Binäre Objekte

Ursache : Das angegebene binäre Objekt konnte nicht gefunden werden.

Der Fehlerwert wird auch dann zurückgegeben, wenn ein binäres Verzeichnis nicht geöffnet werden konnte. Wird ein Problem schon in der Pfadangabe zu dem binären Objekt festgestellt, wird [ErrBinNoPath](#) zurückgegeben.

Kontakt

_ErrBinNoLock

Binäres Objekt nicht gesperrt

Wert -1505

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Binäre Objekte

Ursache : Das binäre Objekt wurde nicht gesperrt. Die Operation kann aber nur bei einem gesperrten Objekt durchgeführt werden.

Kontakt

_ErrBinNoPath

Pfad nicht vorhanden

Wert -1502

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Binäre Objekte

Ursache : Der angegebene Pfad zu dem binären Objekt existiert nicht.

Kontakt

_ErrBinOperation

Operation kann nicht ausgeführt werden

Wert -1510

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Binäre Objekte

Ursache : Es wurde versucht ein binäres Objekt im Wurzelverzeichnis anzulegen. Es wurde versucht ein binäres Objekt zwischen zwei Datenbanken zu kopieren oder zu verschieben.

Die maximale Anzahl von 60 Ebenen wurde überschritten.

Kontakt

Konstanten für Fehler des Druckprozessors

Siehe <u>Alle Befehle</u>	
<u>ErrPpcAcrobat</u>	Fehler beim Erzeugen eines PDF-Dokuments
<u>ErrPpcArgument</u>	Argumente fehlerhaft
<u>ErrPpcDriver</u>	Falscher Treiber
<u>ErrPpcFileCreate</u>	Fehler beim Erzeugen der Datei
<u>ErrPpcFileOpen</u>	Fehler beim Öffnen der Datei
<u>ErrPpcFileRead</u>	Fehler beim Lesen
<u>ErrPpcFileWrite</u>	Fehler beim Schreiben
<u>ErrPpcGhostscript</u>	Fehler beim Erzeugen eines PDF-Dokuments
<u>ErrPpcPrint</u>	Fehler beim Drucken

Kontakt

_ErrPpcAcrobat

Fehler beim Erzeugen eines PDF-Dokuments Wert -
9074

Verwandte Befehle,

Siehe PpcMakeAcrobatPdf(),

ErrGet()

Kategorie : Druckprozessor

Ursache : Beim Erzeugen eines PDF-Dokuments über den Acrobat Distiller ist ein Fehler aufgetreten. Der Distiller schreibt ein eigenes Fehlerprotokoll, dass über die Ursache Aufschluss geben kann.

Kontakt

_ErrPpcArgument

Argumente fehlerhaft

Wert -9034

Verwandte Befehle,

PpcMakePreviewBmp(),

Siehe PpcMakePreviewJpg(),

PpcMakePreviewPng(),

PpcMakePreviewTif(),

ErrGet()

Kategorie : Druckprozessor

Ursache : Ein oder mehrere der angegebenen Argumente sind nicht korrekt.

Kontakt

_ErrPpcDriver

Falscher Treiber

Wert -9070

Verwandte Befehle,

Siehe [PpcMakePdf\(\)](#),

[PpcMakeAcrobatPdf\(\)](#),

[ErrGet\(\)](#)

Kategorie : Druckprozessor

Ursache : Es wurde der Druckjob eines falschen Druckers verwendet, um ein PDF-Dokument zu erzeugen. Ein PDF-Dokument kann nur von einem Postscript-Druckjob, wie ihn der CONZEPT 16-PDF-Drucker erzeugt, generiert werden.

Kontakt

_ErrPpcFileCreate

**Fehler beim Erzeugen der Datei
Wert -9054**

Verwandte Befehle,
PpcMakePreviewBmp(),
PpcMakePreviewJpg(),

Siehe PpcMakePreviewPng(),
PpcMakePreviewTif(),
PpcMakeTif(),
ErrGet()

Kategorie : Druckprozessor

Ursache : Beim Erzeugen der externen Vorschau-Datei ist ein Fehler aufgetreten.

Kontakt

_ErrPpcFileOpen

Fehler beim Öffnen der Datei

Wert **-9051**

[Verwandte Befehle](#),

[PpcMakePreviewBmp\(\)](#),

[PpcMakePreviewJpg\(\)](#),

Siehe [PpcMakePreviewPng\(\)](#),

[PpcMakePreviewTif\(\)](#),

[PpcMakeTif\(\)](#),

[ErrGet\(\)](#)

Kategorie : Druckprozessor

Ursache : Beim Öffnen der externen Vorschau-Datei ist ein Fehler aufgetreten.

Kontakt

_ErrPpcFileRead

Fehler beim Lesen

Wert -9052

Verwandte Befehle,
PpcMakePreviewBmp(),
PpcMakePreviewJpg(),
Siehe **PpcMakePreviewPng()**,
PpcMakePreviewTif(),
PpcMakeTif(),
ErrGet()

Kategorie : Druckprozessor

Ursache : Beim Lesen der externen Vorschau-Datei ist ein Fehler aufgetreten.

Kontakt

_ErrPpcFileWrite

Fehler beim Schreiben

Wert -9053

Verwandte Befehle,

PpcMakePreviewBmp(),

PpcMakePreviewJpg(),

Siehe PpcMakePreviewPng(),

PpcMakePreviewTif(),

PpcMakeTif(),

ErrGet()

Kategorie : Druckprozessor

Ursache : Beim Schreiben der externen Vorschau-Datei ist ein Fehler aufgetreten.

Kontakt

_ErrPpcGhostscript

**Fehler beim Erzeugen eines PDF-Dokuments Wert -
9015**

Siehe [Verwandte Befehle](#),
[PpcMakePdf\(\)](#),
[ErrGet\(\)](#)

Kategorie : Druckprozessor

**Ursache : Beim Erzeugen eines PDF-Dokuments über Ghostscript ist ein Fehler
aufgetreten.**

Diese Fehlermeldung wird zum Beispiel dann zurückgegeben, wenn bei den PpcMake...-Befehlen
eine Datei auf einem Netzwerkpfad erzeugt werden soll. Da der Druckprozessor als Dienst läuft, hat
er in der Regel keinen Zugriff auf verbundene Laufwerke.

Kontakt

_ErrPpcPrint

Fehler beim Drucken

Wert -9081

Verwandte

Siehe Befehle,

PpcPrint(),

ErrGet()

Kategorie : Druckprozessor

Ursache : Beim Drucken des Druckjobs ist ein Fehler aufgetreten.

Kontakt

Konstanten für Fehler bei der Verarbeitung von PDF-Dokumenten

Siehe [Alle Befehle](#)

<u>ErrPdfImageFormat</u>	Ungültiges Format
<u>ErrPdfInsertMetaFile</u>	Fehler beim Einfügen von Informationen
<u>ErrPdfNotLicensed</u>	Erstellung eines PDF-Dokuments mit einer Standard- oder Advanced-Edition
<u>ErrPdfNotPdfA</u>	Fehler bei der Erzeugung eines PDF/A-Dokuments
<u>ErrPdfPageAppend</u>	Seite konnte nicht hinzugefügt werden
<u>ErrPdfPageClosed</u>	Keine Seite zum Bearbeiten geöffnet
<u>ErrPdfPageNotExist</u>	Seite nicht vorhanden
<u>ErrPdfPassword</u>	Ungültiges Kennwort

Kontakt

_ErrPdfImageFormat

Ungültiges Format

Wert -2501

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : PDF-Verarbeitung

Ursache : Das angegebene Bild hat ein ungültiges Format.

Kontakt

_ErrPdfInsertMetaFile

Fehler beim Einfügen von Informationen

Wert -2506

Verwandte

Siehe [Befehle](#),

[PrtJobClose\(\)](#),

[PdfInsertMeta\(\)](#)

Kategorie : PDF-Verarbeitung

Ursache : Es ist ein Fehler beim Einfügen der Informationen in ein PDF-Dokument aufgetreten.

Kontakt

_ErrPdfNotLicensed

Erstellung eines PDF-Dokuments mit einer Standard- oder Advanced-Edition Wert -2508

Verwandte

Siehe [Befehle](#),

[PrtJobClose\(\)](#)

Kategorie : PDF-Verarbeitung

Ursache : Es wurde ein PDF-Dokument mit einer CONZEPT 16-Standard- oder Advance-Edition erstellt. Das Dokument wurde gedruckt, aber mit dem Wasserzeichen "DRAFT" versehen. Zum Erstellen eines PDF-Dokuments wird bis zur Version 5.7.06 eine Enterprise-Edition benötigt (siehe auch [Lizenztypen](#)).



Ab der Version 5.7.06 tritt dieser Fehler nicht mehr auf.

Kontakt

_ErrPdfNotPdfA

Fehler bei der Erzeugung eines PDF/A-Dokuments Wert -

2507

Verwandte

Siehe Befehle,

PrtJobClose()

Kategorie : PDF-Verarbeitung

Ursache : Es konnte kein PDF/A-Dokument erzeugt werden, weil nicht alle
Informationen in das Dokument eingebettet werden konnten.

Kontakt

_ErrPdfPageAppend

Seite konnte nicht hinzugefügt werden

Wert -2505

Verwandte

Siehe Befehle,

PrtJobClose()

Kategorie : PDF-Verarbeitung

Ursache : Bei der Erstellung eines PDF-Dokuments konnte eine Seite nicht erzeugt werden.

Kontakt

_ErrPdfPageClosed

Keine Seite zum Bearbeiten geöffnet

Wert -2503

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : PDF-Verarbeitung

Ursache : Die Anweisung erwartet eine zum Bearbeiten geöffnete Seite eines PDF-Dokuments.

Kontakt

_ErrPdfPageNotExisting

Seite nicht vorhanden

Wert -2504

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : PDF-Verarbeitung

Ursache : Die bei der Anweisung angegebene Seite des PDF-Dokuments ist nicht vorhanden.

Kontakt

_ErrPdfPassword

Ungültiges Kennwort

Wert -2502

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : PDF-Verarbeitung

Ursache : Beim Öffnen eines PDF-Dokuments wurde ein falsches Kennwort angegeben.

Kontakt

Konstanten für Fehler bei regulären Ausdrücken

Alle

Siehe Befehle

[ErrRegExBadEscapeSequence](#)

Nicht aufgelöste Escape-Sequenz im Ausdruck

[ErrRegExBadInterval](#)

Fehler im {min,max} Intervall

[ErrRegExInvalidBackRef](#)

Rückbezug auf eine nicht vorhandene Referenz

[ErrRegExInvalidFlag](#)

Ungültiger Modus

[ErrRegExInvalidRange](#)

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

[ErrRegExLookBehindLimit](#)

Rückschau Ausdrücke müssen eine beschränkte

maximale Länge haben

[ErrRegExMaxLtMin](#)

Im Intervall {min,max} ist max kleiner als min

[ErrRegExMismatchedParentheses](#) Falsch verschachtelte Klammern im regulären

Ausdruck

[ErrRegExMissingCloseBracket](#)

Fehlende schließende Klammer in einem

[ErrRegExNotSupported](#)

Klammerausdruck

[ErrRegExNumberTooBig](#)

Verwendung einer Funktion, die nicht unterstützt wird

[ErrRegExOctalTooBig](#)

Dezimalzahl zu groß

[ErrRegExPropertySyntax](#)

Oktale Zahl muss kleiner oder gleich 0377 sein

[ErrRegExRuleSyntax](#)

Ungültige Unicode-Eigenschaft

[ErrRegExSetContainsString](#)

Syntaxfehler im regulären Ausdruck

[ErrRegExStackOverflow](#)

Reguläre Ausdrücke können keine UnicodeSets mit Zeichenketten beinhalten

[ErrRegExTimeout](#)

Stapelüberlauf in der Ablaufverfolgung des

regulären Ausdrucks

Maximale Suchzeit überschritten

Kontakt

_ErrRegExBadEscapeSequence

Nicht aufgelöste Escape-Sequenz im Ausdruck Wert -

2703

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Nicht aufgelöste Escape-Sequenz im Ausdruck

_ErrRegExBadInterval

Fehler im {min,max} Intervall

Wert -2708

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Fehler im {min,max} Intervall

Kontakt

_ErrRegExInvalidBackRef

Rückbezug auf eine nicht vorhandene Referenz Wert -

2710

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Rückbezug auf eine nicht vorhandene Referenz

_ErrRegExInvalidFlag

Ungültiger Modus

Wert -2711

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Ungültiger Modus

Kontakt

_ErrRegExInvalidRange

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

2716

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Reguläre Ausdrücke

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

Kontakt

[_ErrRegExLookBehindLimit](#)

Rückschau Ausdrücke müssen eine beschränkte maximale Länge haben Wert -2712

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Rückschau Ausdrücke müssen eine beschränkte maximale Länge haben

Kontakt

_ErrRegExMaxLtMin

Im Intervall {min,max} ist max kleiner als min Wert -

2709

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Im Intervall {min,max} ist max kleiner als min

Kontakt

[_ErrRegExMismatchedParentheses](#)

Falsch verschachtelte Klammern im regulären Ausdruck Wert -
2706

[Verwandte](#)

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Falsch verschachtelte Klammern im regulären Ausdruck

Kontakt

_ErrRegExMissingCloseBracket

Fehlende schließende Klammer in einem Klammerausdruck Wert -
2715

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Fehlende schließende Klammer in einem Klammerausdruck

Kontakt

_ErrRegExNotSupported

Verwendung einer Funktion, die nicht unterstützt wird Wert -
2705

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Verwendung einer Funktion, die nicht unterstützt wird

_ErrRegExNumberTooBig

Dezimalzahl zu groß

Wert -2707

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Dezimalzahl zu groß

Kontakt

_ErrRegExOctalTooBig

Oktale Zahl muss kleiner oder gleich 0377 sein Wert -
2714

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Oktale Zahl muss kleiner oder gleich 0377 sein

_ErrRegExPropertySyntax

Ungültige Unicode-Eigenschaft

Wert -2704

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Ungültige Unicode-Eigenschaft

Kontakt

_ErrRegExRuleSyntax

Syntaxfehler im regulären Ausdruck

Wert -2702

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Syntaxfehler im regulären Ausdruck

Kontakt

_ErrRegExSetContainsString

Reguläre Ausdrücke können keine UnicodeSets mit Zeichenketten beinhalten Wert -2713

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Reguläre Ausdrücke können keine UnicodeSets mit Zeichenketten beinhalten

Kontakt

_ErrRegExStackOverflow

Stapelüberlauf in der Ablaufverfolgung des regulären Ausdrucks Wert -2717

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Reguläre Ausdrücke

Bedeutung : Stapelüberlauf in der Ablaufverfolgung des regulären Ausdrucks

Kontakt

_ErrRegExTimeout

Maximale Suchzeit überschritten

Wert -2718

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Reguläre Ausdrücke

Bedeutung : Maximale Suchzeit überschritten

Kontakt

Konstanten für Fehler der ODBC-Schnittstelle

Siehe [Alle Befehle](#)

<u>ErrOdbcEnvironment</u>	ODBC-Umgebung konnte nicht initialisiert werden
<u>ErrOdbcError</u>	Ausführung eines Statements fehlgeschlagen
<u>ErrOdbcFunctionFailed</u>	Funktionsaufruf fehlgeschlagen
<u>ErrOdbcIncomplete</u>	Bibliothek unvollständig
<u>ErrOdbcNoData</u>	Keine (weiteren) Daten vorhanden
<u>ErrOdbcNotFound</u>	Bibliothek nicht geladen
<u>ErrOdbcWarning</u>	Warnung bei der Ausführung eines Statements

Kontakt

_ErrOdbcEnvironment

ODBC-Umgebung konnte nicht initialisiert werden

Wert -553

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : ODBC-Schnittstelle

Ursache : Die ODBC-Umgebung konnte nicht initialisiert werden.

Kontakt

_ErrOdbcError

Ausführung eines Statements fehlgeschlagen Wert -

555

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : ODBC-Schnittstelle

Dieser Fehler wird zurückgegeben, wenn bei der Ausführung eines ODBC-Statements ein Fehler aufgetreten ist. Genauere Informationen

Ursache : über den Fehler können über die OdbcErr...-Eigenschaften des entsprechenden Objekts ermittelt werden.

Kontakt

_ErrOdbcFunctionFailed

Funktionsaufruf fehlgeschlagen

Wert **-554**

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : ODBC-Schnittstelle

Dieser Fehlerwert wird von ODBC-Befehle zurückgegeben, wenn ein

Ursache : Objekt nicht erzeugt werden konnte.

Kontakt

_ErrOdbcIncomplete

Bibliothek unvollständig

Wert -552

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : ODBC-Schnittstelle

Es sind nicht alle notwendigen Funktionen in der dynamischen

Ursache : Link-Bibliothek (odbc32.dll) vorhanden.

Kontakt

_ErrOdbcNoData

Keine (weiteren) Daten vorhanden

Wert -557

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : ODBC-Schnittstelle

Dieser Fehlerwert wird zurückgegeben, wenn keine oder keine weiteren

Ursache : Daten mehr vorhanden sind.

Kontakt

_ErrOdbcNotFound

Bibliothek nicht geladen

Wert -551

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : ODBC-Schnittstelle

Ursache : Die dynamische Link-Bibliothek (odbc32.dll) konnte nicht geladen werden.

Kontakt

_ErrOdbcWarning

Warnung bei der Ausführung eines Statements Wert -

556

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : ODBC-Schnittstelle

Dieser Fehlerwert wird zurückgegeben, wenn bei der Ausführung eines ODBC-Statements eine Warnung aufgetreten ist. Genauere Informationen

Ursache : können über die **OdbcErr...-Eigenschaften des entsprechenden Objekts** ermittelt werden.

Kontakt

Konstanten für Fehler der Benutzerverwaltung Siehe

Alle Befehle

ErrUrmObjectNotFound Objekt nicht gefunden

ErrUrmParentNotFound Elternobjekt nicht gefunden

Kontakt

_ErrUrmObjectNotFound

Objekt nicht gefunden

Wert -2301

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Benutzerverwaltung

Ursache : Das angegebene Objekt wurde nicht gefunden.

Kontakt

_ErrUrmParentNotFound

Elternobjekt nicht gefunden

Wert -2302

Verwandte

Siehe [Befehle](#),

[ErrGet\(\)](#)

Kategorie : Benutzerverwaltung

Ursache : Das angegebene Eltern-Objekt ist nicht mehr vorhanden.

Dieser Fehler kann nur auftreten, wenn zwischen dem Öffnen des Eltern-Objekts und der Verarbeitung eines anderen Befehls, bei dem das Eltern-Objekt als Referenz angegeben ist, das Eltern-Objekt gelöscht wurde. Die Löschung kann durch eine gleichzeitig verarbeitete Prozedur auf dem gleichen Client oder durch die Prozedur eines anderen Clients erfolgen. Der Befehl kann durch das Fehlen des Eltern-Objekts nicht mehr durchgeführt werden.

Konstanten für Fehler bei XML-Befehlen

Siehe [Alle Befehle](#)

[ErrXmlFatal](#) Kritischer Fehler

[ErrXmlNotValid](#) Schema ungültig

[ErrXmlRecoverable](#) Behebbarer Fehler

[ErrXmlWarning](#) Warnung

Kontakt

_ErrXmlWarning

Warnung

Wert -2401

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : XML-Verarbeitung

Bei der Verarbeitung der XML-Datei ist eine Warnung aufgetreten. Ursache :

Weitere Informationen über die Warnung können mit der Anweisung

XmlError() ermittelt werden.

Kontakt

_ErrXmlRecoverable

Behebbarer Fehler

Wert -2402

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : XML-Verarbeitung

Bei der Verarbeitung der XML-Datei ist ein behebbarer Fehler Ursache : aufgetreten. Weitere Informationen über den Fehler können mit der Anweisung **XmlError()** ermittelt werden.

Kontakt

_ErrXmlFatal

Kritischer Fehler

Wert -2403

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : XML-Verarbeitung

Bei der Verarbeitung der XML-Datei ist ein kritischer Fehler aufgetreten. Ursache :

Weitere Informationen über den Fehler können mit der Anweisung

XmlError() ermittelt werden.

Kontakt

_ErrXmlNotValid

Schemadatei ungültig

Wert -2404

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : XML-Verarbeitung

Bei der Verarbeitung der XML-Datei ist die Schemadatei ungültig. Ursache :

Weitere Informationen über den Fehler können mit der Anweisung

XmlError() ermittelt werden.

Kontakt

Konstanten für Fehler der Validierungselemente

Siehe <u>Alle Befehle</u>	
<u>ErrVldExists</u>	<u>Validierungselement existiert bereits</u>
<u>ErrVldLocked</u>	<u>Das Validierungselement ist gesperrt</u>
<u>ErrVldNameInvalid</u>	<u>Ungültiger Name</u>
<u>ErrVldNoFile</u>	<u>Validierungselement nicht vorhanden</u>
<u>ErrVldNoLock</u>	<u>Validierungselement nicht gesperrt</u>

Kontakt

_ErrVldExists

Validierungselement existiert bereits

Wert -1507

Verwandte

Siehe Befehle,

VldOpen(),

ErrGet()

Kategorie : Validierungselemente

Ursache : Es existiert bereits ein Validierungselement mit dem gleichen Namen.

Kontakt

_ErrVldLocked

Das Validierungselement ist gesperrt

Wert -1506

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Validierungselemente

Ursache : Das Validierungselement ist von einem anderen Benutzer gesperrt. Die Operation kann aber nur dann durchgeführt werden, wenn das Element nicht oder vom eigenen Benutzer gesperrt ist.

Kontakt

_ErrVldNameInvalid

Ungültiger Name

Wert -1501

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Validierungselemente

Ursache : Der angegebene Name ist ungültig. In Namen von **Validierungselementen** dürfen keine Steuerzeichen oder die Zeichen "*" und "?" vorkommen.

Kontakt

_ErrVldNoFile

Validierungselement nicht vorhanden

Wert -1503

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Validierungselemente

Ursache : Das angegebene Validierungselement konnte nicht gefunden werden.

Kontakt

_ErrVldNoLock

Validierungselement nicht gesperrt

Wert -1505

Verwandte

Siehe **Befehle**,

ErrGet()

Kategorie : Validierungselemente

Ursache : Das **Validierungselement** wurde nicht gesperrt. Die Operation kann aber nur bei einem gesperrten Element durchgeführt werden.

Weitere Fehlerkonstanten

Siehe **Alle Befehle**

ErrPrtPaperFormat Falsches Papierformat

ErrRtfSyntaxError Syntax Fehler beim Lesen vom RTF-Format

Kontakt

_ErrPrtPaperFormat

Falsches Papierformat

Wert -1601

Verwandte

Siehe Befehle,

PrtJobClose(),

ErrGet()

Kategorie : Drucken

Ursache : Das im Druckjob angegebene Papierformat wird durch die Ausgabeeinheit nicht unterstützt.

_ErrRtfSyntaxError

Syntax Fehler beim Lesen vom RTF-Format

Wert -1701

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : RTF-Verarbeitung

Ursache : Es existiert ein syntaktischer Fehler in der RTF-Datei.

Der Fehler kann beim Laden oder Speichern einer Datei im RTF-Format auftreten.

Kontakt

Liste aller Prozedurbefehle

Liste aller Prozedurbefehle

Siehe **Befehlsgruppen**

A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - R - S - T - U - V - W - X

- [Abs](#)
- [Acos](#)
- [and](#)
- [Asin](#)
- [Atan](#)

- [BinClose](#)
- [BinCopy](#)
- [BinDelete](#)
- [BinDirDelete](#)
- [BinDirMove](#)
- [BinDirOpen](#)
- [BinDirRead](#)
- [BinExport](#)
- [BinImport](#)
- [BinMove](#)
- [BinOpen](#)
- [BinReadMem](#)
- [BinRename](#)
- [BinUpdate](#)
- [BinWriteMem](#)
- [break](#)

- [Call](#)
- [CallOld](#)
- [Ceil](#)
- [ChartClose](#)
- [ChartDataAdd](#)
- [ChartDataClose](#)
- [ChartDataOpen](#)
- [ChartDataSort](#)
- [ChartOpen](#)
- [ChartSave](#)
- [ClipboardRead](#)
- [ClipboardWrite](#)
- [CnvAB](#)
- [CnvAD](#)
- [CnvAF](#)
- [CnvAI](#)
- [CnvAM](#)
- [CnvAT](#)
- [CnvBA](#)
- [CnvBC](#)

- [CnvBF](#)
- [CnvBI](#)
- [CnvBM](#)
- [CnvCB](#)
- [CnvDA](#)
- [CnvDI](#)
- [CnvFA](#)
- [CnvFB](#)
- [CnvFI](#)
- [CnvFM](#)
- [CnvIA](#)
- [CnvIB](#)
- [CnvID](#)
- [CnvIF](#)
- [CnvIL](#)
- [CnvIM](#)
- [CnvIT](#)
- [CnvLI](#)
- [CnvMA](#)
- [CnvMB](#)
- [CnvMF](#)
- [CnvMI](#)
- [CnvTA](#)
- [CnvTI](#)
- [ColorMake](#)
- [ColorRgbMake](#)
- [ComCall](#)
- [ComCallResult](#)
- [ComClose](#)
- [ComEvtProcessGet](#)
- [ComEvtProcessSet](#)
- [ComInfo](#)
- [ComOpen](#)
- [ComPropGet](#)
- [ComPropGetText](#)
- [ComPropSet](#)
- [ComPropSetText](#)
- [Cos](#)
- [CteClear](#)
- [CteClose](#)
- [CteDelete](#)
- [CteInfo](#)
- [CteInsert](#)
- [CteInsertItem](#)
- [CteInsertNode](#)
- [CteNodeValueAlpha](#)
- [CteOpen](#)
- [CteRead](#)
- [cycle](#)

- DateDay
- DateDayOfWeek
- DateMake
- DateMonth
- DateYear
- DbaConnect
- DbaControl
- DbaDisconnect
- DbaInfo
- DbaKeyRebuild
- DbaLicense
- DbaLog
- DbaName
- DbgConnect
- DbgControl
- DbgDisconnect
- DbgDump
- DbgStop
- DbgTrace
- DdeCommand
- DdeConnect
- DdeDisconnect
- DdeGetData
- DdeGetDataInfo
- DdeInit
- DdeServiceClose
- DdeServiceData
- DdeServiceOpen
- DdeServiceRead
- DdeSetData
- DdeTerm
- Dec
- declare
- define
- div
- DllCall
- DllLoad
- DllUnload
- do...while
- DtaBegin
- DtaCommit
- DtaRollback

- ErrCall
- ErrGet
- ErrIgnore
- ErrMapText
- ErrPos
- ErrSet
- ErrThrowProc

- [ErrTryCatch](#)
- [ErrTryIgnore](#)
- [exit](#)
- [Exp](#)

- [FileInfo](#)
- [FileInfoByName](#)
- [FileName](#)
- [Fld...](#)
- [Fld...ByName](#)
- [FldAttributes](#)
- [FldCompare](#)
- [FldCopy](#)
- [FldDef](#)
- [FldDefByName](#)
- [FldInfo](#)
- [FldInfoByName](#)
- [FldName](#)
- [Floor](#)
- [FontMake](#)
- [for...loop...while/until](#)
- [Fract](#)
- [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](#)
- [FsiTime](#)
- [FsiWrite](#)
- [FsiWriteMem](#)

- [global](#)

- [handle](#)
- [HdlEnum](#)
- [HdlInfo](#)
- [HdlLink](#)
- [HttpClose](#)
- [HttpGetData](#)
- [HttpOpen](#)

- [if...else](#)
- [Inc](#)

- [JobClose](#)
- [JobControl](#)
- [JobEvent](#)
- [JobOpen](#)
- [JobSleep](#)
- [JobStart](#)
- [JsonLoad](#)
- [JsonSave](#)

- [KeyFldInfo](#)
- [KeyInfo](#)
- [KeyInfoByName](#)
- [KeyName](#)

- [LinkFldInfo](#)
- [LinkInfo](#)
- [LinkInfoByName](#)
- [LinkName](#)
- [local](#)
- [LocaleLoad](#)
- [LocaleSelect](#)
- [LocaleUnload](#)
- [Log10](#)
- [Log2](#)

- LogN

- MailClose
- MailData
- MailOpen
- main
- Max
- MemAllocate
- MemCnv
- MemCompress
- MemCopy
- MemDecrypt
- MemEncrypt
- MemFindByte
- MemFindStr
- MemFree
- MemGenKeyPair
- MemHash
- MemHMAC
- MemReadByte
- MemReadStr
- MemSign
- MemUncompress
- MemVerify
- MemWriteByte
- MemWriteStr
- Min
- Modulo-Operator
- MsxClose
- MsxOpen
- MsxRead
- MsxReadMem
- MsxWrite
- MsxWriteMem

- NetInfo

- OdbcCatalogClm
- OdbcCatalogTbl
- OdbcClmData
- OdbcClmInfo
- OdbcClose
- OdbcConnect
- OdbcConnectDriver
- OdbcDataSources
- OdbcExecute
- OdbcExecuteDirect
- OdbcFetch

- [OdbcOpen](#)
- [OdbcParamAdd](#)
- [OdbcParamSet](#)
- [OdbcPrepare](#)
- [OEMLoad](#)
- [OEMSavE](#)
- [or](#)

- [PdfAttachExportFile](#)
- [PdfAttachExportMem](#)
- [PdfAttachFile](#)
- [PdfAttachInfoGet](#)
- [PdfAttachMem](#)
- [PdfClose](#)
- [PdfInsertImage](#)
- [PdfInsertMeta](#)
- [PdfNew](#)
- [PdfOpen](#)
- [PdfPageClose](#)
- [PdfPageOpen](#)
- [PdfSavePage](#)
- [PdfTextColor](#)
- [PdfTextExtractMem](#)
- [PdfTextFont](#)
- [PdfTextWrite](#)
- [PicExifInfo](#)
- [PointMake](#)
- [Pow](#)
- [PpcMakeAcrobatPdf](#)
- [PpcMakeEps](#)
- [PpcMakePdf](#)
- [PpcMakePreviewBmp](#)
- [PpcMakePreviewJpg](#)
- [PpcMakePreviewPng](#)
- [PpcMakePreviewTif](#)
- [PpcMakeTif](#)
- [PpcPrint](#)
- [ProcAdvertise](#)
- [ProcCacheClear](#)
- [ProcCompile](#)
- [PrtAdd](#)
- [PrtAddByName](#)
- [PrtDeviceClose](#)
- [PrtDeviceOpen](#)
- [PrtDeviceRefresh](#)
- [PrtFormClose](#)
- [PrtFormOpen](#)
- [PrtInfo](#)
- [PrtInfoStr](#)
- [PrtJobClose](#)

- [PrtJobOpen](#)
- [PrtJobWrite](#)
- [PrtPrinterRefresh](#)
- [PrtPropGet](#)
- [PrtPropSet](#)
- [PrtRtfSearch](#)
- [PrtSearch](#)
- [PrtUnit](#)
- [PrtUnitLog](#)

- [Random](#)
- [RangeMake](#)
- [RecBufClear](#)
- [RecBufCompare](#)
- [RecBufCompareFld](#)
- [RecBufCopy](#)
- [RecBufCreate](#)
- [RecBufDefault](#)
- [RecBufDestroy](#)
- [RecDelete](#)
- [RecDeleteAll](#)
- [RecFilterAdd](#)
- [RecFilterCreate](#)
- [RecFilterDestroy](#)
- [RecInfo](#)
- [RecInsert](#)
- [RecLink](#)
- [RecLinkInfo](#)
- [RecRead](#)
- [RecReplace](#)
- [RectMake](#)
- [Rescale](#)
- [return](#)
- [RmtCall](#)
- [RmtDataRead](#)
- [RmtDataSearch](#)
- [RmtDataWrite](#)
- [Rnd](#)
- [RtfTabMake](#)

- [SbrClear](#)
- [SbrCompare](#)
- [SbrCopy](#)
- [SbrInfo](#)
- [SbrInfoByName](#)
- [SbrName](#)
- [SbrStatus](#)
- [SckClose](#)
- [SckConnect](#)

- [SckInfo](#)
- [SckListen](#)
- [SckRead](#)
- [SckReadMem](#)
- [SckStartTls](#)
- [SckWrite](#)
- [SckWriteMem](#)
- [SelAddLink](#)
- [SelAddSortFld](#)
- [SelClear](#)
- [SelClose](#)
- [SelCopy](#)
- [SelCreate](#)
- [SelDefQuery](#)
- [SelDelete](#)
- [SelIgnore](#)
- [SelInfo](#)
- [SelInfoAlpha](#)
- [SelInfoDate](#)
- [SelInfoTime](#)
- [SelOpen](#)
- [SelRead](#)
- [SelRecDelete](#)
- [SelRecInsert](#)
- [SelRun](#)
- [SelStore](#)
- [SelValue](#)
- [Sgn](#)
- [Sin](#)
- [Sqrt](#)
- [StoClose](#)
- [StoDirOpen](#)
- [StoDirRead](#)
- [StoExport](#)
- [StoImport](#)
- [StoImportTile](#)
- [StoOpen](#)
- [StoReadMem](#)
- [StoWriteMem](#)
- [StoWriteTileMem](#)
- [StrAdj](#)
- [StrChar](#)
- [StrCnv](#)
- [StrCut](#)
- [StrDecrypt](#)
- [StrDel](#)
- [StrEncrypt](#)
- [StrFind](#)
- [StrFmt](#)
- [StrIns](#)
- [StrLen](#)

- [StrToChar](#)
- [sub](#)
- [SvcSessionControl](#)
- [switch...case...default](#)
- [SysBeep](#)
- [SysDate](#)
- [SysExecute](#)
- [SysGetArg](#)
- [SysGetEnv](#)
- [SysOS](#)
- [SysPropGet](#)
- [SysPropSet](#)
- [SysSleep](#)
- [SysTics](#)
- [SysTime](#)
- [SysTimerClose](#)
- [SysTimerCreate](#)

- [Tan](#)
- [TapiCall](#)
- [TapiClose](#)
- [TapiConference](#)
- [TapiConferenceCommit](#)
- [TapiConferenceDial](#)
- [TapiDevice](#)
- [TapiDial](#)
- [TapiForward](#)
- [TapiInfo](#)
- [TapiListen](#)
- [TapiOpen](#)
- [TapiPickup](#)
- [TextClear](#)
- [TextClose](#)
- [TextCopy](#)
- [TextCreate](#)
- [TextDelete](#)
- [TextEdit](#)
- [TextInfo](#)
- [TextInfoAlpha](#)
- [TextInfoDate](#)
- [TextInfoTime](#)
- [TextLineRead](#)
- [TextLineWrite](#)
- [TextOpen](#)
- [TextRead](#)
- [TextRename](#)
- [TextSearch](#)
- [TextWrite](#)
- [TimeHour](#)
- [TimeHSec](#)

- TimeMake
- TimeMin
- TimeSec
- Trn
- try
- trysub

- UrmClose
- UrmCreate
- UrmDelete
- UrmOpen
- UrmPermGet
- UrmPermGetRaw
- UrmPermSet
- UrmPropGet
- UrmPropSet
- UrmPropType
- UrmRead
- UserClear
- UserID
- UserInfo
- UserName
- UserNumber
- UserPassword

- VarAllocate
- VarCopy
- VarFree
- VarInfo
- VarInstance
- VarName
- VldClose
- VldDelete
- VldDirOpen
- VldDirRead
- VldOpen
- VldUpdate

- while
- WinAdd
- WinAddByName
- WinBarcodeSaveImage
- WinBeep
- WinBoxScrollVisible
- WinClose
- WinColorOpacityGet
- WinColorOpacitySet
- WinCopy

- [WinCreate](#)
- [WinCroNavigate](#)
- [WinCroPrint](#)
- [WinCroReload](#)
- [WinCroSelection](#)
- [WinDestroy](#)
- [WinDialog](#)
- [WinDialogBox](#)
- [WinDialogResult](#)
- [WinDialogRun](#)
- [WinDocLoadBin](#)
- [WinDocLoadName](#)
- [WinDocSaveBin](#)
- [WinDocSaveName](#)
- [WinDocUserDictAddName](#)
- [WinDocUserDictRemoveName](#)
- [WinEmfProcess](#)
- [WinEvtProcessGet](#)
- [WinEvtProcessSet](#)
- [WinEvtProcNameGet](#)
- [WinEvtProcNameSet](#)
- [WinFlash](#)
- [WinFocusGet](#)
- [WinFocusSet](#)
- [WinGanttBoxAdd](#)
- [WinGanttCellInfo](#)
- [WinGanttIvlAdd](#)
- [WinGanttIvlRemove](#)
- [WinGanttLineAdd](#)
- [WinHalt](#)
- [WinIconPreload](#)
- [WinInfo](#)
- [WinLayer](#)
- [WinLstCellGet](#)
- [WinLstCellSet](#)
- [WinLstDatLineAdd](#)
- [WinLstDatLineInfo](#)
- [WinLstDatLineRemove](#)
- [WinLstEdit](#)
- [WinMenuContext](#)
- [WinMenuItemAdd](#)
- [WinMenuItemRemove](#)
- [WinMsdDelete](#)
- [WinMsdDeleteName](#)
- [WinMsdInsert](#)
- [WinMsdInsertName](#)
- [WinMsdRead](#)
- [WinMsdReadName](#)
- [WinMsdUpdate](#)
- [WinOpen](#)
- [WinPropGet](#)

- [WinPropSet](#)
- [WinRemove](#)
- [WinRtfLoad](#)
- [WinRtfLoadBin](#)
- [WinRtfLoadName](#)
- [WinRtfPicInsertMem](#)
- [WinRtfPicInsertName](#)
- [WinRtfSave](#)
- [WinRtfSaveBin](#)
- [WinRtfSaveName](#)
- [WinRtfSearch](#)
- [WinRtfTabGet](#)
- [WinRtfTabSet](#)
- [WinRvwColumn](#)
- [WinRvwEdit](#)
- [WinRvwUpdate](#)
- [WinSave](#)
- [WinSearch](#)
- [WinSearchClear](#)
- [WinSearchPath](#)
- [WinSearchPathGet](#)
- [WinShutdownBlock](#)
- [WinSleep](#)
- [WinThemeClose](#)
- [WinThemeDelete](#)
- [WinThemeOpen](#)
- [WinThemeSetDelete](#)
- [WinThemeSetOpen](#)
- [WinTreeNodeAdd](#)
- [WinTreeNodeRemove](#)
- [WinTreeNodeSearch](#)
- [WinUpdate](#)
- [WinUrmDialog](#)
- [WinUserEvent](#)
- [with](#)
- [WseArg](#)
- [WseInfo](#)
- [WseReturn](#)
- [WseStatus](#)
- [WseStrCnv](#)

- [XmlClose](#)
- [XmlError](#)
- [XmlGetValueAlpha](#)
- [XmlGetValueInt](#)
- [XmlLoad](#)
- [XmlOpenReader](#)
- [XmlOpenWriter](#)
- [XmlRead](#)
- [XmlSave](#)

Kontakt

- XmlWrite
- xor

Kontakt

Deskriptoren

Verwendung von Deskriptoren

Alle Objekte werden in CONZEPT 16 über einen Deskriptor identifiziert. Die Deskriptoren der Objekte werden durch die entsprechenden Anweisungen zurückgegeben. Die Deskriptoren der untergeordneten Objekte können über die Anweisung WinInfo() bzw. PrtInfo() ermittelt werden.

Alle Deskriptoren mit Ausnahme von Storage- und Binären Objekten verfügen über folgende Eigenschaften:

- Name
- ID
- Custom

Die Eigenschaften können vom Programmierer verwendet werden. Insbesondere können die entsprechenden Deskriptoren in dynamischen Strukturen organisiert werden.

Bei allen Prozedurbefehlen die einen Deskriptor als erstes Argument erwarten, kann der Deskriptor auch vor den Befehl geschrieben und mit dem Operator -> an den Befehl übergeben werden.

Deskriptoren für Oberflächen- und Druckobjekte können über den Namen des Objektes ermittelt werden. Dazu wird dem Objektnamen das \$-Zeichen vorangestellt.

Zur Ermittlung des Deskriptors zu dem Namen wird der komplette Objekt-Baum verwendet. Die Suche wird dabei von einem bestimmten Objekt innerhalb des Suchbaums gestartet. Das Objekt ist entweder das zuletzt geladenen Fenster-Objekt oder das Objekt, das das Ereignis ausgelöst hat. Zunächst wird das Start-Objekt überprüft und anschließend alle untergeordneten Objekte. Konnte das Objekt nicht gefunden werden, wird das Fenster-Objekt und dessen untergeordnete Objekte durchsucht. Die Suche wird beendet, sobald das erste Objekt mit dem entsprechenden Namen gefunden wird.

Das Startobjekt bestimmt, wie lange die Suche dauert und welche Objekte durchsucht werden. Der Suchpfad wird automatisch gesetzt, wenn ein Ereignis ausgelöst oder ein Frame-Objekt geladen wird. Alle Namensreferenzen werden dann innerhalb des Fenster-Objekts aufgelöst.

Mit dem Befehl WinSearchPath() wird das Startobjekt der Suche definiert. Wurde innerhalb eines Ereignisses ein Objekt mit dem Befehl WinOpen() mehrfach geladen, ist der Suchpfad zunächst auf das zuletzt geladene Fenster gesetzt. Da die Namen der Objekte nicht mehr eindeutig sind, muss, um ein Objekt aus dem zuerst geladenen Fenster anzusprechen, der Suchpfad auf dieses Fenster gesetzt werden.



Die Suche wird bei jeder Namensreferenz erneut durchgeführt. Wird ein Deskriptor innerhalb einer Funktion mehrfach verwendet (zum Beispiel in einer Schleife), sollte der Deskriptor in einer Variablen gespeichert und diese als Objektreferenz genutzt werden.

Beispiele:

Kontakt

```
// Eingabeobjekt "edName" inkativ setzen$edName->wpDisabled # true;// Selektion ausführenHdlSel
```

Deskriptoren von Fenster- bzw. Druckobjekten lassen sich ebenfalls über folgende Befehle ermitteln:

- WinInfo()
- WinSearch()
- PrtInfo()
- PrtSearch()

In jedem Ereignis kann der Deskriptor des ereignisauslösenden Objektes über das Argument aEvt ermittelt werden.

Mit dem Befehl HdlInfo() können über einen beliebigen Deskriptor Informationen eingeholt werden. Über HdlLink() lassen sich Deskriptoren miteinander verknüpfen. Durch HdlEnum() lassen sich alle Deskriptoren enumerieren.

Kontakt

Oberflächen-Objekte

Ein Dialog besteht aus ein oder mehrerer Objekte.

[Alphabetische Liste](#)

Siehe [aller](#)

[Oberflächen-Objekte](#)

Die Objekte untergliedern sich in folgende Bereiche:

- [Form-Objekte](#)
- [Eingabe-Objekte](#)
- [Schaltflächen-Objekte](#)
- [Ansicht-Objekte](#)
- [Ausgabe-Objekte](#)
- [Toolbar-Objekte](#)
- [Anordnung-Objekte](#)
- [COM-Objekte](#)
- [Weitere Objekte](#)

Kontakt

Application



Application

Siehe [App](#), [Liste](#),
[Eigenschaften](#)

Die Eigenschaften des Application-Objekts wirken sich auf die Oberfläche der gesamten Applikation aus. Das Objekt wird vom System generiert. Es wird nicht in der Entwicklungsumgebung angelegt. Über den Befehl [WinInfo\(\)](#) mit dem Parameter [WinApplication](#) oder das Schlüsselwort [App](#) wird das Application-Objekt ermittelt.

Über die Eigenschaften dieses Objekts können unter anderem die installierten Systemdrucker ermittelt und applikationsweite Eigenschaften gesetzt werden.

Die Einstellungen in diesem Objekt bleiben bis zum Ende der Applikation erhalten, können aber zur Laufzeit geändert werden. Werden die Einstellungen in einer Prozedur geändert, die aus dem Designer über den Menüpunkt "Datei / Testen" aufgerufen werden, bleiben sie bis zum Ende der Prozedur erhalten.

Das Application-Objekt steht nur Prozeduren und Funktionen zur Verfügung, die vom CONZEPT 16-Client (Standard oder Advanced) durchgeführt werden. Der Zugriff auf das Objekt zum Beispiel durch eine Prozedur, die durch die [CONZEPT 16-Programmierschnittstelle](#) oder den [SOA-Service](#) ausgeführt wird ist nicht möglich.

Form-Objekte

Siehe Alle Oberflächen-Objekte

Frame



AppFrame



MdiFrame



TrayFrame



PrintDoc



PrintDocRecord



PrintForm



PrintFormList



Kontakt

Frame



Anwendungsfenster

Liste,

Siehe Eigenschaften,

Ereignisse,

Dialogerstellung

Das Anwendungsfenster ist die Benutzeroberfläche für den Anwender.

Für die Erstellung einer MDI-Anwendung stehen die Fenstertypen Applikationsfenster und MDI-Anwendungsfenster zur Verfügung.

Damit bestehende MdiFrame-Objekte auch als Frame-Objekte benutzt werden können, besteht die Möglichkeit ein MdiFrame-Objekt in ein Frame-Objekt zu konvertieren. Hierzu existiert der Menüeintrag Konvertieren zu Frame im Kontextmenü des MdiFrame-Objekts, sowie der Menüeintrag Bearbeiten / Konvertieren zu Frame.

Der Titel des Anwendungsfensters kann mit der Eigenschaft Caption gesetzt werden.

Kontakt

AppFrame



Applikationsfenster

Liste,

Siehe Eigenschaften,
Ereignisse

Das Applikationsfenster ist das Rahmenfenster einer MDI (multiple document interface)-Anwendung. Es kann nur Fenster vom Typ MdiFrame aufnehmen.

Ein Applikationsfenster besitzt im Arbeitsbereich keine Interaktionselemente. Ein Applikationsfenster sollte immer ein Menü besitzen. Wird ein neues Applikationsfenster erstellt und der Menüeditor gestartet, wird ein Standardmenü vorgeschlagen, welches bereits Menüeinträge für die Darstellung und die Organisation der untergeordneten MdiFrame-Objekte enthält.

Der Titel des Anwendungsfensters kann mit der Eigenschaft Caption gesetzt werden.



Wird die Eigenschaft StyleTheme auf WinStyleThemeModern gesetzt, sollte dem AppFrame ein ToolbarMenu-Objekt hinzugefügt werden. Sonst werden die Minimieren- und Schließen-Schaltflächen der MDI-Fenster nicht angezeigt.

MdiFrame

MDI-Anwendungsfenster

Liste,

Eigenschaften,

Ereignisse,

Siehe [Dialogerstellung](#) [Ereignisabläufe](#).

MdiFrame,

Datensatzpuffer

(Blog)

Ein Applikationsfenster kann ein oder mehrere untergeordnete MdiFrame-Objekte aufnehmen und darstellen (siehe [WinAdd\(\)](#) und [WinAddByName\(\)](#)). Zwar werden MdiFrame-Objekte im Designer unabhängig vom AppFrame-Objekt entworfen, brauchen jedoch zur Laufzeit immer ein AppFrame-Objekt.

Jedes Unterfenster kann innerhalb der Anwendung mehrfach mit unterschiedlichen oder auch identischen Daten erscheinen. Das MDI-Fenster kann den Arbeitsbereich des Applikationsfensters belegen und ist gleichzeitig durch dessen Größe begrenzt.

Wird die maximale Größe des Unterfensters gewählt, wird der gesamte Arbeitsbereich des Applikationsfensters genutzt. Als Name der Anwendung erscheint der Titel des Applikationsfensters und Titel des jeweiligen Unterfensters getrennt durch einen Bindestrich in der Titelzeile des Applikationsfensters.

MdiFrame-Objekte können alle Objekte der Objektpalette aufnehmen. Jedoch kann ihnen kein Menü zugewiesen werden. Lediglich das Applikationsfenster besitzt ein Menü. Das Menü muss ebenfalls die entsprechenden Steuerungselemente für das MDI-Fenster enthalten.

Zum Testen der MdiFrame-Objekte existiert wie gewohnt der Testmodus. Da die Objekte immer innerhalb eines Applikationsfensters laufen, wird für den Testmodus ein Standard-Applikationsfenster erzeugt.

Damit bestehende Frame-Objekte auch als MdiFrame-Objekte benutzt werden können, besteht die Möglichkeit ein Frame-Objekt in ein MdiFrame-Objekt zu konvertieren. Hierzu existiert der Menüeintrag Konvertieren zu MdiFrame im Kontextmenü des Frame-Objekts, sowie der Menüeintrag Bearbeiten / Konvertieren zu MdiFrame.

Wird der Menüeintrag ausgewählt, wird ein neues MdiFrame-Objekt erzeugt und die Objekte des Frame-Objekts übertragen. Hier bleibt zu beachten, dass ein evtl. vorhandenes Menü nicht mit übertragen werden kann, da MdiFrame-Objekte kein Menü besitzen können.

Der Titel des Anwendungsfensters kann mit der Eigenschaft Caption gesetzt werden.



Da in einer MDI-Anwendung ein Fenster mehrfach dargestellt werden kann (z. B. Dialog "Kunden" wurde mehrmals mit unterschiedlichen Daten geladen) muss für jeden Dialog ein eigener Datensatzpuffer angelegt werden. Über die Eigenschaft DbRecBuf kann einem MdiFrame ein Datensatzpuffer zugeordnet werden. Die Verwaltung des Puffers geschieht automatisch.

Kontakt

Wird ein MdiFrame-Objekt mehrfach geladen, kann es bei der Referenzierung auf Objekte innerhalb des Fensters über den Namen des Objekts zu Schwierigkeiten kommen, da der Name nicht mehr eindeutig ist. Die Namensauflösung funktioniert korrekt, solange innerhalb der Ereignisse des MDI-Fensters auf Objekte des gleichen Fensters referenziert wird. Sollen Objekte des anderen MDI-Fensters angesprochen werden, muss der Suchpfad entsprechend gesetzt werden (siehe [WinSearchPath\(\)](#)).

Die Auflösung von Objekt-Referenzen (\$Name) in Prozeduren kann auf das aktive MDI-Fenster beschränkt werden. Dazu muss die Ausprägung [WinAppSearchMdiFrame](#) der [Flags](#)-Eigenschaft des [App](#)-Objektes gesetzt werden. Die Laufzeiten für die Referenzauflösung werden damit verkürzt, besonders bei Referenzen, die nicht aufgelöst werden können.



Wird die Eigenschaft [StyleTheme](#) auf [WinStyleThemeModern](#) gesetzt, sollte dem AppFrame ein [ToolbarMenu](#)-Objekt hinzugefügt werden. Sonst werden die Minimieren- und Schließen-Schaltflächen der MDI-Fenster nicht angezeigt.

TrayFrame

 **Tray-Anwendungsfenster**
Liste,

Siehe Eigenschaften,
Ereignisse,
Dialogerstellung

Das TrayFrame-Objekt dient der Erstellung von Tray-Anwendungen. Solche Anwendungen werden im SysTray-Bereich der Windows-Taskbar dargestellt. Folgende Komponenten sind für eine TrayFrame-Anwendung mindestens notwendig:

- Icon

Über die Eigenschaft muss ein Icon zugewiesen werden, welches nach dem Starten der TrayFrame-Anwendung im SysTray-Bereich dargestellt wird.

- MenuNameCntxt

Über die Eigenschaft MenuNameCntxt muss ein Menü zugewiesen werden. Das Menü wird angezeigt, wenn der Benutzer mit der rechten Maustaste auf das Icon klickt. Ein Doppelklick auf das Icon führt den fett hervorgehobenen Menüeintrag aus, der durch die Eigenschaft Default = true des MenuItems konfiguriert ist.

Die Tray-Anwendung wird mit der Anweisung WinDialog() oder WinDialogRun() gestartet.

Kontakt

PrintDoc



[PrintDoc](#)

[Liste](#),

Siehe [Eigenschaften](#),

[PrtInfo\(\)](#),

[Blog](#)

Das PrintDoc-Objekt (**Druckdokument**) definiert eine druckbare Vorlage für eine oder mehrere Seiten mit beliebigem Inhalt. Das Objekt sollte immer dann verwendet werden, wenn alle Informationen innerhalb der Seite auf festen Positionen zu finden sind. Zum Beispiel in Briefvorlagen oder Formularen (Überweisungsträger usw.).

Das Druckdokument kann nur **Seiten**-Objekte aufnehmen. Es enthält Eigenschaften, die das Papierformat ([PageFormat](#)) und die Ausrichtung ([Orientation](#)) der Seiten bestimmen.

Die Seitenobjekte selbst enthalten wiederum Druckobjekte, die den Seiteninhalt bestimmen. Wird ein PrintDoc-Objekt im Designer erstellt, sind in der Objektpalette nur die Registerreiter "Druck" und "Seite" vorhanden.

Das PrintDoc-Objekt kann bis zu zehn Seiten-Objekte aufnehmen. Der Druckjob selbst kann mehr Seiten umfassen, da einzelne Seiten aus dem PrintDoc-Objekt mehrfach (mit unterschiedlichen Inhalten) gedruckt werden können.

In einem Druckjob ([PrtJobOpen\(\)](#)) kann das PrintDoc-Objekt über die Funktion [PrtInfo\(\)](#) mit der Option [PrtDoc](#) ermittelt werden.

Kontakt

PrintDocRecord



[PrintDocRecord](#)

[Liste](#),

Siehe [Eigenschaften](#),

[Blog](#)

Das PrintDocRecord-Objekt (Tabellen-Dokument) ist ein spezielles [PrintDoc](#)-Objekt. Es kann lediglich eine Seite beinhalten. Wie beim PrintDoc-Objekt, definiert das Objekt ein Papierformat ([PageFormat](#)), sowie die Ausrichtung ([Orientation](#)) der Seite.

Die im Tabellen-Dokument enthaltene Seite kann wiederum nur ein Tabellenobjekt aufnehmen. Durch diese Kombination von Druckdokument, Seite und Tabelle ist die einfache Erstellung von Drucklisten möglich.

Ein PrintDocRecord-Objekt kann bei der Erstellung im Designer mit dem Tabellen-Assistent konfiguriert werden. Wurde das Objekt erstellt, sind in der Objekt-Palette die Registerreiter "Druck", "Seite" und "Tabelle" vorhanden.

Die angegebene Anzahl von Spalten werden beim Erzeugen gleichmäßig über die gesamte Seitenbreite angeordnet. Die Breite der Spalten kann entweder mit der Maus oder über die Eigenschaft Width verändert werden. Wird beim Ändern der Breite

einer Spalte mit der Maus die -Taste gedrückt, werden alle übereinander liegenden Spalten mit verändert.

In der Tabelle werden alle Datensätze einer Datei in der Reihenfolge des angegebenen Schlüssels ausgegeben. Die Anzahl der Datensätze kann über eine Selektion ([DbSelection](#)) oder Filter ([DbFilter](#)) zur Laufzeit eingeschränkt werden.

PrintForm



PrintForm

[Liste](#),

[Eigenschaften](#),

Siehe [Beispiel](#),

[PrintFormList](#),

[Blog](#)

Das PrintForm-Objekt dient dem dynamischen Aufbau von Druckdokumenten ([PrintDoc](#)-Objekt). Hierzu gibt es spezielle Prozedurbefehle, die die im PrintForm-Objekt enthaltenen Druckobjekte einem PrintDoc-Objekt hinzufügen.

Das PrintForm-Objekt wird dann verwendet, wenn ein Druckjob aus kleineren oder sehr unterschiedlichen Elementen aufgebaut werden soll. Zum Beispiel eine Rechnung oder eine Liste, die auf Daten aus mehreren Tabellen zugreift.

Wird ein PrintForm-Objekt im Designer erstellt, ist in der Objektpalette nur der Registerreiter "Druck" vorhanden, und nur die dort enthaltenen Objekte können dem Objekt hinzugefügt werden.

Kontakt

PrintFormList



PrintFormList
Liste,
Siehe Eigenschaften,
PrintForm,
Blog

Das PrintFormList-Objekt dient der Erstellung von Drucklisten. Es handelt sich bei dem Objekt um einen Formular-Typ, analog zu einem PrintForm-Objekt, jedoch mit spezieller Funktionalität.

Dem PrintFormList-Objekt können in der ersten Ebene nur Zeilenobjekte (PrtLine-Objekte) hinzugefügt werden. Die PrintFormList richtet die Zeilenobjekte so aus, dass eine Zeilenstruktur (also eine Liste) entsteht.

Das Zeilenobjekt kann für optische Zwecke oder zur Erzeugung eines Zeilenabstandes mit einem Rahmenobjekt (PrtLineBorder) versehen werden. Die Zeilenobjekte können ebenfalls ausgeblendet werden, wenn kein Inhalt dargestellt wird (Eigenschaft SkipPrint).

Die Breite aller Zeilenobjekte definiert sich über die Eigenschaft Width des PrintFormList-Objekts. Außerdem haben alle Zeilenobjekte einen Abstand vom linken Rand der PrintFormList (MarginLeft) und einen Abstand vom oberen Rand (MarginTop).

Eingabe-Objekte

Objekte der Eingabe-Palette

Siehe Alle Oberflächen-Objekte

Edit 

IntEdit 

BigIntEdit 

FloatEdit 

DecimalEdit 

TimeEdit 

DateEdit 

ColorEdit 

FontNameEdit 

FontSizeEdit 

TextEdit 

RtfEdit 

CodeEdit 

Kontakt

Edit

 **Eingabeobjekt für einzeilige Zeichenketten**

Liste,

Siehe Eigenschaften,

Ereignisse,

alpha

Der Inhalt des Eingabeobjekts kann mit der Eigenschaft Caption gesetzt und gelesen werden.

In der Eigenschaft DbFieldName kann ein Datenbankfeld von beliebigem Typ mit dem Eingabeobjekt verknüpft werden. Ist in den Einstellungen die Option "Setzen von DbFieldName setzt auch Eingabelimit" aktiviert, wird die Eigenschaft LengthMax auf die Länge des Datenbankfeldes gesetzt, wenn zuvor in der Eigenschaft kein kleinerer Wert eingetragen war.

Für eine mehrzeilige Eingabe von Zeichenketten stehen die Objekte TextEdit und RtfEdit zur Verfügung.

Kontakt

IntEdit

 **Eingabeobjekt für ganzzahlige Werte**
Liste,

Siehe Eigenschaften,
Ereignisse, int

Der Inhalt des Eingabeobjekts kann mit der Eigenschaft CaptionInt gesetzt und gelesen werden. Die zulässigen Werte liegen im Bereich von -2.147.483.647 und +2.147.483.647.

In der Eigenschaft DbFieldName kann ein Datenbankfeld mit dem Eingabeobjekt verknüpft werden. Einem IntEdit-Objekt kann auch ein Datenbankfeld vom Typ Ganzahlig kurz (word) zugewiesen werden. Ist in den Einstellungen des Designers die Option "Setzen von DbFieldName setzt auch Eingabelimit" aktiviert, werden die Eigenschaften MinInt und MaxInt auf die Beschränkungen des Datenbankfeldes gesetzt, wenn zuvor in der Eigenschaft keine engeren Grenzen eingetragen waren.

Die Eigenschaft InputCtrl bestimmt, ob die Eingabehilfe aktiv ist.

BigIntEdit



Eingabeobjekt für große ganze Zahlen

Liste,

Siehe Eigenschaften,

Ereignisse,

bigint

Der Inhalt des Eingabeobjekts kann mit der Eigenschaft CaptionBigInt gesetzt und gelesen werden. Die zulässigen Werte liegen im Bereich von -9.223.372.036.854.775.807 und +9.223.372.036.854.775.807.

In der Eigenschaft DbFieldName kann ein Datenbankfeld mit dem Eingabeobjekt verknüpft werden. Einem BigIntEdit-Objekt können auch Datenbankfelder der Datentypen Ganzzahlig kurz (word) und Ganzzahlig lang (int) zugewiesen werden. Ist in den Einstellungen des Designers die Option "Setzen von DbFieldName setzt auch Eingabelimit" aktiviert, werden die Eigenschaften MinBigInt und MaxBigInt auf die Beschränkungen des Datenbankfeldes gesetzt, wenn zuvor in der Eigenschaft keine engeren Grenzen eingetragen waren.

Kontakt

FloatEdit

 **Eingabeobjekt für Gleitkommazahlen**
Liste,

Siehe Eigenschaften,
Ereignisse,
float

In diesem Objekt können Werte vom Typ float eingegeben werden. Der Inhalt des Eingabeobjekts kann mit der Eigenschaft CaptionFloat gesetzt und gelesen werden.

In der Eigenschaft DbFieldName kann ein Datenbankfeld vom Typ float (Gleitkomma) mit dem Eingabeobjekt verknüpft werden.

Die Eigenschaft InputCtrl bestimmt, ob die Eingabehilfe aktiv ist.

DecimalEdit



Eingabeobjekt für Dezimalzahlen

Liste,

Siehe Eigenschaften,

Ereignisse,

decimal

Der Inhalt des Eingabeobjekts kann mit der Eigenschaft CaptionDecimal gesetzt und gelesen werden.

In der Eigenschaft DbFieldName kann ein Datenbankfeld vom Typ decimal (dezimal) mit dem Eingabeobjekt verknüpft werden.

Wird die Eigenschaft FmtDecimalFlags in der Ausprägung FmtNumNoZero gesetzt, können die Inhalte "0" und "undefiniert" nicht mehr unterschieden werden. In beiden Fällen wird das Eingabeobjekt leer dargestellt.

Verliert das Eingabeobjekt den Fokus und ist das Eingabeobjekt leer, wird je nach Ausprägung der Eigenschaft DecEditFlags der Wert des Eingabefeldes auf "undefiniert"

(WinDecEditDecimalUndef ist gesetzt) oder "0" (WinDecEditDecimalUndef ist nicht gesetzt) gesetzt.

Kontakt

TimeEdit

 **Eingabeobjekt für Zeitwerte**

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#),

[time](#)

Der Inhalt des Eingabeobjekts kann mit der Eigenschaft [CaptionTime](#) gesetzt und gelesen werden.

In der Eigenschaft [DbFieldName](#) kann ein Datenbankfeld vom Typ [time](#) (Zeit) mit dem Eingabeobjekt verknüpft werden.

Die Eigenschaft [InputCtrl](#) bestimmt, ob die Eingabehilfe aktiv ist.

Der Nullwert für TimeEdit ist 24:00:00.

Kontakt

DateEdit

 **Eingabeobjekt für Datumswerte**
Liste,

Siehe Eigenschaften,
Ereignisse,
date

Der Inhalt des Eingabeobjekts kann mit der Eigenschaft CaptionDate gesetzt und gelesen werden. In der Eigenschaft DbFieldName kann ein Datenbankfeld vom Typ date (Datum) mit dem Eingabeobjekt verknüpft werden.

Die Eigenschaft InputCtrl bestimmt, ob die Eingabehilfe aktiv ist.

Standardmäßig wird bei einer zweistelligen Eingabe des Jahres 1900 zu dem Jahreswert addiert. Die Eingabe 01.01.01 wird demnach als 01.01.1901 interpretiert. Über die Eigenschaft DateWindow lässt sich steuern, ob 1900 oder 2000 zu dem Jahreswert addiert werden soll.

Das eingegebene Datum wird auf Gültigkeit überprüft. Über die Eigenschaft InputCheck wird festgelegt, ob die Überprüfung bereits während der Eingabe oder nach Verlassen des Objektes stattfindet.

Eine Schnelleingabe des Datums kann durch das Setzen der Eigenschaft FormatDate auf den Wert WinFmtDateDDMM erreicht werden. In diesem Fall reicht bereits die Angabe des Tages in zweistelliger Schreibweise. Monat und Jahr werden aus der Eigenschaft DefaultDate ergänzt.

Soll ein bestehender Wert in dem DateEdit-Objekt nur teilweise überschrieben werden, kann in der Eigenschaft Flags die Ausprägung WinAppEditDateFast angegeben werden. Bekommt das Objekt den Eingabefokus, wird nicht der gesamte Inhalt des Objektes selektiert und er kann Zeichenweis überschrieben werden.

Der Nullwert für DateEdit ist 0.0.0.

Durch die Eigenschaft StyleTheme wird beeinflusst, ob die Kalenderanzeige des Objekts in Theme-Darstellung WinStyleThemeSystem oder in den angegebenen Farben WinStyleThemeNone gezeichnet wird. Die Eigenschaft wirkt sich bei dem Objekt nur aus, wenn die gleiche Eigenschaft beim App-Objekt auf WinStyleThemeSystem gesetzt ist.

Das Datum kann mit Hilfe der folgenden Tastaturkommandos relativ zum eingetragenen Datum verändert werden:

Tastenkombination Veränderung relativ zum aktuellen Datum



Gleicher Wochentag der vorherigen Woche



Gleicher Wochentag der nächsten Woche



Vorheriger Tag

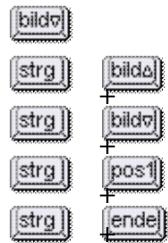


Nächster Tag



Gleicher Tag des vorherigen Monats

Kontakt



- Gleicher Tag des nächsten Monats**
- Gleiches Datum des vorhergehenden Jahres**
- Gleiches Datum des nächsten Jahres**
- 1. Tag des Monats des aktuellen Datums**
- Letzter Tag des Monats des aktuellen Datums**

Ist das Datum der 29.2. eines Schaltjahres und wird **[strg]** und **[bildv]** oder **[bildv]** gedrückt, wird auf den 28.2. des vorhergehenden bzw. nachfolgenden Jahres positioniert.



Die Tastaturkombinationen **[strg] + [bildv]** und **[strg] + [bildv]** können nur verwendet werden, wenn die Eigenschaft **EmulateKeys** des übergeordneten Frame-Objekts auf **false** gesetzt ist.

ColorEdit

 Eingabeobjekt für Farbwerte

Liste,

Siehe Eigenschaften,

Ereignisse

Der Inhalt des Eingabeobjekts kann in der Eigenschaft CaptionColor gesetzt und gelesen werden.

In diesem Objekt können Werte der Datentypen word (ganzzahlig kurz) und int (ganzzahlig lang) dargestellt werden.

Über die Eigenschaft ModeColor kann eingestellt werden, ob nur Standardfarben oder Standard- / System- und benutzerdefinierte Farben auswählbar sind.

Die Eigenschaft InputCtrl bestimmt, ob die Eingabehilfe aktiv ist.

FontNameEdit

 **Objekt zur Auswahl der Schriftart**

Liste,

Siehe Eigenschaften,

Ereignisse

Über dieses Objekt kann eine Auswahl der auf dem System installierten Zeichensätze getroffen werden. Der ausgewählte Zeichensatz kann in der Eigenschaft Caption ausgelesen werden.

Mit der gleichen Eigenschaft kann ebenfalls eine Vorbelegung gesetzt werden.

Für die Größe des Zeichensatzes kann das Objekt FontSizeEdit verwendet werden. Die Verbindung zwischen dem FontNameEdit-Objekt und dem FontSizeEdit-Objekt wird durch die Eigenschaft ObjLink des FontNameEdit-Objekts hergestellt. Die Eigenschaft wird automatisch beim Einfügen der Objekte eingetragen.

Das Objekt FontNameEdit ist auch Bestandteil der Werkzeugleiste vom Typ ToolbarRtf.

Kontakt

FontSizeEdit

 **Objekt zur Auswahl des Schriftgrades**

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#)

Mit diesem Objekt kann der Schriftgrad einer Schriftart ausgewählt werden. Die Schriftart wird in einem [FontNameEdit](#)-Objekt ausgewählt, das mit diesem Objekt über die Eigenschaft [ObjLink](#) verbunden ist. Die Auswahl der Schriftart setzt automatisch die Eigenschaft [FontName](#) des Objekts. Das FontSizeEdit-Objekt stellt nur die Schriftgrößen zur Verfügung, die für diese Schriftart definiert sind.

Das Objekt FontSizeEdit ist auch Bestandteil der Werkzeugleiste vom Typ [ToolbarRtf](#).

Der Schriftgrad kann in der Eigenschaft [CaptionInt](#) gesetzt und gelesen werden.

Kontakt

TextEdit



Eingabeobjekt für mehrzeilige Zeichenketten

Liste,

Siehe Eigenschaften,

Ereignisse,

RtfEdit

Der Inhalt des **TextEdit**-Objekts kann über die Eigenschaften Caption, DbFieldName oder DbTextBuf gesetzt werden. Bei Verwendung eines Textpuffers (DbTextBuf) bleibt das Setzen der Eigenschaften Caption und DbFieldName ohne Wirkung und das Auslesen liefert einen leeren alpha-String.

In der Eigenschaft DbFieldName kann ein Datenbankfeld mit beliebigem Typ angegeben werden.

Befinden sich innerhalb des internen Textes Linienzeichen, kann zur korrekten Darstellung des Textes der Zeichensatz "C16 Courier" oder "C16 Terminal" in der Eigenschaft Font eingetragen werden.

Die Länge des Textes ist nicht limitiert.

Kontakt

RtfEdit

Eingabeobjekt für mehrzeiligen formatierten Text Liste,

Eigenschaften,

Ereignisse,

Siehe TextEdit,

Befehle, Text

und Daten

mischen,

PrtRtf

In diesem Objekt kann unformatierter Text und Text im Rich Text Format (RTF) angezeigt und editiert werden. Der Text kann aus unterschiedlichen Quellen stammen. Die Quelle wird in der Eigenschaft StreamSource angegeben.

In den Eigenschaften des Objekts kann eine generelle Formatierung in Bezug auf die Schriftart (FontName und RtfEffect), Schriftgröße (FontSize und FontSizeTwips) und der Textausrichtung (RtfAlign) vorgenommen werden.

Soll innerhalb des Textes die Formatierung verändert werden, wird eine Toolbar vom Typ ToolbarRtf benötigt. Wird die Toolbar eingefügt, verbindet sie sich automatisch mit einem bereits existierenden RtfEdit-Objekt. Genauso verbindet sich das RtfEdit-Objekt mit einem bereits bestehenden ToobarRtf-Objekt. Bestehen mehrere Möglichkeiten eine Verbindung herzustellen, wird immer eine Verbindung zum ersten Objekt hergestellt.

Die Verbindung wird über die Eigenschaft ObjLink hergestellt.

Die Länge des Textes wird durch seine Quelle limitiert. Wird der Text in der Eigenschaft Caption angegeben, können maximal 8192 Zeichen geschrieben bzw. dargestellt werden.

Wurde eine andere Quelle angegeben, bezieht sich der Inhalt der Eigenschaft Caption auf den selektierten Bereich. Eine Selektion innerhalb des Textes kann über die Eigenschaft Range angegeben werden. Hiermit kann ebenfalls die Einfügemarkierung innerhalb des Textes positioniert werden, wenn der minimale und maximale Wert der Eigenschaft gleich ist. Wird die Eigenschaft Range auf den Wert RangeMake(-1, -1) gesetzt, wird das Ende des Textes angesprungen.

Ein markierter Text kann mit Hilfe von Formatierungsanweisungen formatiert werden.

Im RtfEdit-Objekt werden folgende Tastaturkommandos ausgewertet:

- Navigation

- ♦ Der Cursor kann mit den Pfeiltasten im Text bewegt werden.
- ♦  +  und  +  positioniert den Cursor an den Anfang des folgenden oder des vorigen Wortes.
- ♦  +  und  +  scrollt den Text seitenweise nach unten und oben.
- ♦  und  positioniert innerhalb einer Zeile auf die erste Spalte, oder auf das letzte Zeichen.

Kontakt

- ◆ und **blättert eine Bildschirmseite.**
- ◆ + + **positioniert auf den Anfang bzw. auf das Ende des gesamten Textes.**
- ◆ + und + **positioniert den Cursor an den Anfang bzw. an das Ende des angezeigten Textes.**

• Markieren

- ◆ **Zum Markieren können die Tastenkombinationen zur Navigation bei gedrückter -Taste verwendet werden.**
- ◆ + **markiert den gesamten Text.**
- ◆ + **+ löscht den markierten Text oder von der Cursorposition bis zum Ende des Wortes.**
- ◆ + **löscht den markierten Text oder von der Cursorposition bis zum Anfang des Wortes.**

• Ausschneiden / Kopieren / Einfügen

- ◆ + oder + + **schnidet den markierten Textbereich aus.**
- ◆ + oder + + **kopiert den markierten Textbereich.**
- ◆ + oder + + **fügt den ausgeschnittenen/kopierten Textbereich ein.**
- ◆ + **+ macht die letzte Änderung rückgängig.**
- ◆ + **stellt die letzt rückgängig gemachte Änderung wieder her.**

• Formatierung und Eingabe von Sonderzeichen

- ◆ + **Absatz zentriert.**
- ◆ + **Darstellung im Blocksatz.**
- ◆ + **Absatz rechtsbündig.**
- ◆ + **Absatz linksbündig.**
- ◆ + **tief gestellt.**
- ◆ + **+ hoch gestellt.**
- ◆ + **einfacher Zeilenabstand.**
- ◆ + **doppelter Zeilenabstand.**
- ◆ + **halber Zeilenabstand.**
- ◆ + **Wechsel des Aufzählungszeichen.**
- ◆ + **wandelt den markierten Bereich in Großbuchstaben oder Kleinbuchstaben.**
- ◆ + **Umwandlung des links neben der Einfügemarken stehenden Hexadezimalcodes in das entsprechende Unicode-Zeichen.**
- ◆ + + **Umwandlung des links neben der Einfügemarken stehenden Unicode-Zeichens in den entsprechenden Hexadezimalcode.**

Beispiele:

```
tRange:min # 1; tRange:max # 30;// alternativ tRange # RangeMake(1, 30); $RtfEdit->wpRange # tRang
```

Kontakt

Text und Daten mischen

Texte mit Datensätzen verbinden

Ein bestehender Text oder RTF-Text kann mit dem Inhalt von Datensatzpuffern bzw. von globalen Variablen verbunden werden. Dazu müssen im Text entsprechende Platzhalter eingetragen werden. Die Platzhalter werden dabei in ein Markierungszeichen geklammert, um sie vom normalen Text unterscheiden zu können. Dies kann sowohl zur Erstellung von Serienbriefen als auch zum dynamischen Erzeugen von einzelnen Texten verwendet werden.

Standardmäßig wird die Tilde "~" als Markierungszeichen verwendet. Das Zeichen kann aber im Applikationsobjekt in der Eigenschaft RtfMixMarker für RtfEdit-Objekte und in der Eigenschaft DocMixMarker für CtxDocEdit-Objekte auf ein beliebiges anderes Zeichen geändert werden.

Das Mischen des Textes mit den Daten kann zu verschiedenen Zeitpunkten erfolgen:

- Beim Lesen eines Textes in ein RtfEdit-Objekt

Dazu muss bei den Befehlen (zum Beispiel WinRtfLoad()) die Option WinRtfLoadMix angegeben werden.

- Beim Speichern eines Textes aus einem RtfEdit-Objekt

Dazu muss bei den Befehlen (zum Beispiel WinRtfSave()) die Option WinRtfSaveMix angegeben werden.

- Beim Lesen eines Textes in ein CtxDocEdit-Objekt

Dazu muss bei den Befehlen (zum Beispiel WinDocLoadName()) die Option WinDocLoadMix angegeben werden.

- Beim Speichern eines Textes aus einem CtxDocEdit-Objekt

Dazu muss bei den Befehlen (zum Beispiel WinDocSaveName()) die Option WinDocSaveMix angegeben werden.

- Beim Drucken eines Textes mit dem PrtRtf-Objekt

Ist beim Drucken des Objektes in der Eigenschaft PrtRtfFlags die Ausprägung PrtRtfMix gesetzt, werden die Markierungen durch die entsprechenden Inhalte ersetzt.

Folgende Markierungen werden beim Mischen von Text und Daten ersetzt:

- ~F:<Feldname>~

Als <Feldname> wird der Name eines Feldes aus der Datenstruktur oder die Nummer der Datei, des Teildatensatzes und des Feldes durch Komma getrennt verwendet. Der Platzhalter wird durch den Inhalt des angegebenen Feldes ersetzt. Das Ausgabeformat wird durch die Ländereinstellungen des Betriebssystems bestimmt. Soll ein anderes Ausgabeformat verwendet werden, kann mit den Befehlen LocaleLoad() und LocaleSelect() dieses Format ausgewählt werden. Darüber hinaus können je nach Typ des Feldes noch folgende Formatoptionen angegeben werden:

Kontakt

Feldtyp	Option	Beschreibung
Alphanumerisch '/W1' <u>(alpha)</u>		Erste Wortumstellung: Die Wortumstellungen gehen davon aus, dass die verschiedenen Namensbestandteile (Name, Vorname und Titel) durch Kommata voneinander getrennt in einem Datenbankfeld stehen ('Müller-Lüdenscheidt, Franz, Dr.'). Durch die erste Wortumstellung wird der Name in der Form Titel, Vorname und Name dargestellt. Aus dem Namen im Beispiel wird dadurch 'Dr. Franz Müller-Lüdenscheidt'.
'/W2'		Zweite Wortumstellung: In dieser Wortumstellung wird der Vorname ausgelassen. Das Beispiel wird in 'Dr. Müller-Lüdenscheidt' umgewandelt.
'/W3'		Dritte Wortumstellung: Hier werden keine Doppelnamen angezeigt. Das Beispiel wird in 'Dr. Müller' umgewandelt.
'/Bn'		Ist das Feld nicht leer, werden nach der Ausgabe des Inhaltes noch die in <i>n</i> angegebenen Leerzeichen ausgegeben. Es können bis zu 999 Leerzeichen angegeben werden.
'/Sn'		Das Feld wird mit einer statischen Ausgabelänge ausgegeben. Unabhängig von Inhalt des Feldes werden die in <i>n</i> angegebenen Zeichen ausgegeben. <i>n</i> kann einen Wert von 1 bis 999 angegeben werden. Der Ausgabewert wird abgeschnitten oder mit Leerzeichen aufgefüllt.
Ganzzahlig (<u>byte</u> , '/G' <u>word</u> , <u>int</u> und <u>bigint</u>)		Es findet keine Tausender-Trennung statt.
'/Z'		Ist der darzustellende Wert gleich 0, wird keine Ausgabe erzeugt (Nullunterdrückung).
Gleitkomma (<u>float</u> und <u>decimal</u>)	'/G'	Es findet keine Tausender-Trennung statt.
	'/Z'	Ist der darzustellende Wert gleich 0, wird keine Ausgabe erzeugt (Nullunterdrückung).
	'/P'	Anstelle des Dezimaltrennzeichens wird ein Punkt ausgegeben.
	'/n'	

Kontakt

Datum (date) '/L'

Über diese Option wird die Anzahl der Nachkommastellen definiert. Soll die Anzahl der Nachkommastellen definiert werden, muss entweder zuvor mindestens einer der bereits genannten Formatoptionen oder ein "/" ohne Optionen angegeben werden. Werden keine Nachkommastellen angegeben, werden alle Nachkommastellen angezeigt.

Zeit (time) '/S'
 '/H'

Mit dieser Option wird das lange Datumsformat eingestellt. Wird kein langes Datumsformat gewählt, erscheint das Datum in der kurzen Anzeige.

Logisch (logic)

'/<true>/<false>' Logische Felder und Variablen müssen mit Werten versehen werden, die in Abhängigkeit des Wertes angezeigt werden sollen. Die erste Zeichenkette wird angezeigt, wenn der Wert true ist, die zweite Zeichenkette wird angezeigt, wenn der Wert false ist.

- ~G:<Variablenbereich>:<Variablenname>~

Für Variablen gelten die gleichen Formatierungsmöglichkeiten, wie für Felder.

Es können mehrere Optionen hinter einem "/" gesammelt werden. Bei Zahlenwerten können die Formatierungsanweisungen (bis auf die Anzahl der Nachkommastellen) kombiniert werden: '~F:KND.fNumber/GZ/2~'. Die Anzahl der Nachkommastellen werden in einer zweiten Option angegeben.

Damit die Platzhalter korrekt ersetzt werden können, müssen die Namen korrekt geschrieben sein und es darf kein Formatwechsel zwischen den Markierungszeichen vorgenommen werden.

Markierungen, die nicht ersetzt werden können, bleiben im Text enthalten.

Beispiele:

Platzhalter	ersetzter Text
'~F:KND.aName~'	'Müller, Walter, Dr.'
'~F:KND.aName/W1~'	'Dr. Walter Müller'
'~F:10,1,2/W2~'	'Dr. Müller'
'~F:KND.fNumber~'	'1.256'
'~F:KND.fNumber/G/2~'	'1256,00'
'~F:KND.fNumber//2~'	'1.256,00'
'~F:KND.lActive/ja/nein~' 'nein'	

CodeEdit



Editor für Prozeduren und Texte

Liste,

Siehe Eigenschaften,

Ereignisse,

Befehle

Das CodeEdit-Objekt dient der Eingabe von Prozeduren und Texten und ist Basis für den Editor. Der Inhalt wird über die Eigenschaft FileName definiert. Hierbei können sowohl interne, als auch externe Dokumente angezeigt werden. Das Objekt unterstützt Syntax-Highlighting für CONZEPT 16, XML- und JSON-Dokumente. Der Typ des angezeigten Textes wird über die Eigenschaft EditorTextType definiert. Ist die Eigenschaft EditorFold aktiv, kann der Inhalt des CodeEdit-Objektes zwischen zusammengehörenden Klammern zusammengeklappt werden.



Der Zugriff auf Eigenschaften und Funktionen des CodeEdit-Objektes ist erst ab dem EvtCreated des enthaltenden Elternfensters möglich.

Im CodeEdit-Objekt können folgende Tastenkombinationen verwendet werden:

Navigation

- Der Cursor kann mit den Pfeiltasten im Text bewegt werden.
- + und + positioniert den Cursor an den Anfang des folgenden oder des vorigen Wortes.
- und positioniert innerhalb einer Zeile auf das erste oder letzte Zeichen, das kein Leerzeichen ist. Beim erneuten Drücken der gleichen Taste wird auf das erste oder letzte Zeichen der Zeile positioniert.
 und blättert eine Bildschirmseite.
 + und + positioniert auf den Anfang bzw. auf das Ende des gesamten Textes.
- + , + , + und + scrollt den Inhalt des Editors zeilen- bzw. spaltenweise, ohne den Cursor zu verschieben.
- versieht die aktuelle Zeile mit einem Lesezeichen zu dem zu einem späteren Zeitpunkt mit + bzw. + gesprungen werden kann.
 + ... + + + und + wechselt zwischen der öffnenden und schließenden Klammer, wenn der Cursor auf einer Klammer steht (unabhängig davon, welche Klammern verwendet werden).
- + ... + wechselt zwischen den Bereichen, wenn ein Text in verschiedene Bereiche aufgeteilt wurde.

Markieren

- Zum Markieren können die Tastenkombinationen zur Navigation bei gedrückter -Taste verwendet werden.
- Ein Bereich kann durch Ziehen mit der Maus markiert werden. Soll ein rechteckiger Bereich (nicht zeilenweise) markiert werden, kann die Maus mit gedrückten + - bzw. + -Tasten verwendet werden.
- + markiert den gesamten Text.
- + + markiert einen rechteckigen Bereich.

Kontakt

- markiert einen rechteckigen Bereich (entspricht).
- Sind mehrere Zeilen markiert, können diese durch oder bzw. ausgerückt werden.
- Mit oder kann eine rechteckige Markierung links oder rechts verschoben werden.
- + + oder + + markiert zwischen der öffnenden und schließenden Klammer, wenn der Cursor auf einer Klammer steht (unabhängig davon, welche Klammern verwendet werden).

Ausschneiden / Kopieren / Einfügen

- oder schneidet den markierten Textbereich aus.
- oder kopiert den markierten Textbereich.
- oder fügt den ausgeschnittenen/kopierten Textbereich ein.
- löscht die aktuelle Zeile.

Folding

- einzelnen Bereich zuklappen.
- einzelnen Bereich aufklappen.
- alle Bereiche zuklappen.
- alle Bereiche aufklappen.

Zoom

- oder Ansicht vergrößern.
- oder Ansicht verkleinern.
- + Ansicht zurücksetzen.

Sonstiges

- + löscht den Text bis zum Ende des Wortes.
- + blendet Leerzeichen ein/aus
- + wandelt den markierten Bereich in Großbuchstaben.
- + wandelt den markierten Bereich in Kleinbuchstaben.
- setzt/löscht ein Lesezeichen.
- + springt zum nächsten Lesezeichen.
- + springt zum vorherigen Lesezeichen.
- + aktiviert und deaktiviert die Anzeige der zusammengehörenden Klammern.
- + fügt das aktuelle Datum an der Cursorposition ein.
- + blendet die Anzeige der Zeilennummern ein bzw. aus.
- + kommentiert die markierten Zeilen aus oder entfernt die Kommentarzeichen. Ist mindestens eine Zeile nicht auskommentiert, werden alle Zeilen auskommentiert.

Kontakt

-  +  fügt unabhängig vom automatischen Klammerungsmodus einen Zeilenumbruch ein.

Schaltflächen-Objekte

Siehe Alle Oberflächen-Objekte

Button 

MenuButton 

ColorButton 

RadioButton 

CheckBox 

Kontakt

Button

 **Button**
Liste,
Siehe Eigenschaften,
Ereignisse

Durch das Betätigen einer Schaltfläche werden unterschiedliche Aktionen ausgelöst.

Folgende Aktionen sind möglich:

- Ereignis EvtClicked

Die hier angegebene Prozedur wird ausgeführt.

Über das Kontextmenü der Schaltfläche lassen sich Aktionen definieren, deren Ausführung keiner Prozedur bedürfen:

- Schließen

Die Eigenschaft TypeButton wird auf den Wert WinBtnClose gesetzt. Die Eigenschaft ID erhält den Wert WinIdClose. Das Drücken der Schaltfläche führt zu einem Schließen des Fensters.

- Abbrechen

Die Eigenschaft TypeButton wird auf den Wert WinBtnClose gesetzt. Die Eigenschaft ID erhält den Wert WinIdCancel. Das Drücken der Schaltfläche führt zu einem Schließen des Fensters.

- Ok

Die Eigenschaft ID erhält den Wert WinIdOk. Das Drücken der Schaltfläche führt zu einem Schließen des Fensters.

- Hilfe

Die Eigenschaft TypeButton wird auf den Wert WinBtnHelp gesetzt. Die Eigenschaft ID erhält den Wert WinIdHelp. Das Drücken der Schaltfläche ruft die in der Eigenschaft Help angegebene Hilfe auf.

- RTF-Toolbar

Für die Verwendung des Buttons in einer Toolbar für das RtfEdit-Objekt, stehen verschiedene Funktionen zur Auswahl:

Fett Kursiv
Unterstrichen
Linksbündig
Rechtsbündig Zentriert
Blocksatz
Aufzählungszeichen
Textfarbe
Hintergrundfarbe

Kontakt

Das Auslösen einer Schaltfläche kann mit der Maus oder über die Tastatur erfolgen. Bei Verwendung der Tastatur wird über die Leertaste die Schaltfläche ausgelöst. Soll ein Drücken auch über die Return-Taste möglich sein, muss entweder die Eigenschaft EmulateKeys des Fensters auf false gesetzt sein, andernfalls findet ein Wechsel des Eingabefokus auf das nächste Objekt statt, oder die Eigenschaft ReturnKeyClick hat den Wert true. In diesem Fall kann der Button immer mit der Return-Taste aktiviert werden, sofern er den Fokus besitzt.

Eine Schaltfläche kann ebenfalls durch eine mnemonische Auswahl aktiviert werden. Dazu wird in der Eigenschaft Caption dem gewünschte Auswahlzeichen das &-Zeichen vorangestellt. Das Auswahlzeichen wird dann mit einem Unterstrich dargestellt und der Button kann über die Kombination  + Auswahlzeichen ausgelöst werden.

Neben der Beschriftung kann ein Button auch über ein Symbol verfügen. In der Eigenschaft ImageTile kann aus einer Reihe von vordefinierten Symbolen gewählt werden. Das Verwenden eigener Symbole ist ebenfalls möglich. Dazu muss eine entsprechende Grafik in die Datenbank importiert werden. In der Eigenschaft ImageTileUser wird dann die Nummer der zu verwendenden Kachel angegeben.



Um eine Drag & Drop-Operation auf einem Button auszulösen, muss zusätzlich die  -Taste gedrückt werden.

Kontakt

MenuButton



MenuButton

Liste,

Siehe Eigenschaften,

Ereignisse

Das Objekt MenuButton funktioniert identisch zum Button-Objekt. Lediglich beim

Anklicken der Pfeilschaltfläche oder Drücken der Tastenkombination + bzw.

+ wird ein Menü angezeigt. Das Menü wird in der Eigenschaft MenuName eingetragen.

Vor dem Laden des Menüs wird das Ereignis EvtMenuPopup aufgerufen. Hier kann das zu ladende Menü geändert werden. In dem anschließend aufgerufenen Ereignis EvtMenuInitPopup können noch Änderungen an den einzelnen Menüeinträgen vorgenommen werden, bevor das Menü angezeigt wird.

Wird ein Menüpunkt ausgewählt, wird das Ereignis EvtMenuCommand aufgerufen. Das Ereignis wird sowohl bei der Auswahl eines Menüeintrages aus dem Kontextmenü als auch bei der Auswahl eines Eintrages aus dem Popupmenü aufgerufen. Der Deskriptor des geladenen Menüs kann mit der Anweisung WinInfo() ermittelt werden.

Kontakt

ColorButton

 Eingabeobjekt für Farbwerte

Liste,

Siehe Eigenschaften,

Ereignisse

Über dieses Objekt kann der Benutzer Farben auswählen. Die ausgewählte Farbe kann über die Eigenschaft CaptionColor gesetzt und gelesen werden.

Kontakt

Radiobutton

Einfachauswahlknopf

Liste,

Siehe Eigenschaften,

Ereignisse

Auswahlknöpfe sind geeignet um den Benutzer zwischen verschiedenen festgelegten Eingaben wählen zu lassen. Von allen Auswahlknöpfen die dasselbe Vaterobjekt haben kann jeweils nur ein Knopf ausgewählt sein.

Der Status des Knopfes kann mit der Eigenschaft CheckState gelesen und gesetzt werden.

Wird ein Dialog aufgerufen, bekommt das erste Objekt in dem Dialog den Fokus. Im Falle eines Radiobuttons bedeutet das automatisch, dass der erste Radiobutton ausgewählt wird. Soll der Fokus auf den aktvierten Radiobutton gesetzt werden, muss die Ausprägung WinAppRadioFocusChecked der Eigenschaft Flags des App-Objekts gesetzt sein.

Kontakt

Checkbox



Mehrfachauswahlknopf

Liste,

Siehe Eigenschaften,

Ereignisse,

logic

Dieses Objekt kann dazu verwendet werden, Einstellungen von Parametern, die entweder ein oder aus sein müssen, anzuzeigen. Im Gegensatz zu dem Radiobutton handelt es sich um einen Mehrfachauswahlknopf.

In diesem Objekt kann nur der Datentyp logic dargestellt werden.

Der Status des Häkchens kann mit der Eigenschaft CheckState gelesen und gesetzt werden.

Objekte zur Datenansicht

Siehe Alle Oberflächen-Objekte

DataList



StoList



RecList



RecView



TreeView



DataSet



DataSet

[Liste](#),
[Eigenschaften](#),

Siehe [Ereignisse](#),
[Befehle](#),
[Fragen zur](#)
[DataSet](#)

Mit diesem Objekt können Informationen zeilenweise angezeigt werden. Das DataSet-Objekt ähnelt im Aufbau dem RecList-Objekt. In dem DataSet-Objekt werden allerdings keine Datensätze aus der Datenbank sondern Daten, die sich im Hauptspeicher befinden, dargestellt. Der Datentyp der darzustellenden Informationen wird über die Eigenschaft ClmType spezifiziert. Die Eigenschaft kann zur Laufzeit nur gelesen werden. Der Typ der Spalte wird somit beim Erstellen des DataSet-Objekts festgelegt. Über die Eigenschaften TileNameUser und ClmTypeImage können Grafiken in einer DataSet angezeigt werden.

Die darzustellenden Informationen werden zur Laufzeit der Liste zugeordnet. Einzelne Zeilen werden mit dem Befehl WinLstDatLineAdd() der Liste hinzugefügt oder mit WinLstDatLineRemove() entfernt.

Die Informationen innerhalb einer Zeile werden mit WinLstCellSet() gesetzt und mit WinLstCellGet() gelesen.

Zum Bearbeiten der Liste im Designer steht der DataSet-Editor zur Verfügung.

Bei diesem Objekt ist eine Mehrfachselektion möglich. Zur Verwendung der Mehrfachselektion, muss die Eigenschaft MultiSelect gesetzt werden. Weitere Informationen befinden sich im Abschnitt SelectionData-Objekt.

Column

 Spalteneintrag

Liste,
Eigenschaften
Spalte /
RecList,
Eigenschaften
Spalte /
Siehe RecView,
Eigenschaften
Spalte /
DataList,
Eigenschaften
Spalte /
StoList

Die Objekte RecList, RecView, DataList und StoList bestehen aus einem oder mehreren Spalteneinträgen.

Kontakt

DataListPopup



DataListPopup

Liste,

Siehe Eigenschaften,
Ereignisse

Mit diesem Objekt wird das DataList-Objekt als Combobox dargestellt. In diesem Fall ist die DataList ein Unterobjekt des PopupList-Objekts.

Mit diesem Objekt können Informationen zeilenweise angezeigt werden. Das DataListPopup-Objekt ähnelt im Aufbau dem RecListPopup-Objekt. In dem DataListPopup-Objekt werden allerdings keine Datensätze aus der Datenbank sondern Daten, die sich im Hauptspeicher befinden, dargestellt. Der Datentyp der darzustellenden Informationen wird über die Eigenschaft ClmType spezifiziert. Die Eigenschaft kann zur Laufzeit nur gelesen werden. Der Typ der Spalte wird somit beim Erstellen des DataList-Objekts festgelegt. Über die Eigenschaften TileNameUser und ClmTypeImage können Grafiken in einem DataListPopup-Objekt angezeigt werden.

Eine DataListPopup wird standardmäßig mit maximal 10 Zeilen dargestellt. Über die Eigenschaft MaxLines kann eine Anzahl zwischen 1 und 100 Zeilen gesetzt werden.

Neben der Maus kann die Liste über die Tastenkombination + oder per Prozedur über die Eigenschaft PopupOpen aufgeklappt werden.

Für die Bearbeitung einer DataListPopup stehen entsprechende Prozedurbefehle zur Verfügung.

StoList



Liste mit Datenbankelementen

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#),

[Storage-Objekte](#)

In diesem Objekt können alle vom Programmierer angelegten Ressourcen der Applikation angezeigt werden. Innerhalb einer Liste kann immer nur eine Art von Ressource angezeigt werden. Dies wird in der Eigenschaft [StoView](#) bestimmt.

Ausgehend von einem [Benutzer](#), einer [Benutzergruppe](#) oder einer [Elementgruppe](#) können auch alle Mitgliedschaften bzw. alle zugeordneten Elemente in der Liste angezeigt werden. Dazu muss der Name des Benutzers, der Benutzergruppe oder der Elementgruppe in der Eigenschaft [UrmParentName](#) angegeben werden. Es werden dann die Benutzergruppen des Benutzers, die Mitglieder einer Benutzergruppe bzw. die Elemente einer Elementgruppe angezeigt. Zur Laufzeit kann auch der Deskriptor des entsprechenden Objekts in die Eigenschaft [UrmParentHandle](#) eingetragen werden.

Die Spalten dieses Objekts sind fest vorgegeben und können auch nicht aus der Liste entfernt werden. Sollen bestimmte Informationen nicht angezeigt werden, müssen diese über die Eigenschaft [Visible](#) unsichtbar gesetzt werden. Die Reihenfolge der Spalten kann beliebig verändert werden.

Folgende Spalten werden erzeugt:

- ClmIcon - Symbol des Typs
- ClmName - Name des Objekts
- ClmSize - Größe des Objekts in Bytes
- ClmLines - Anzahl der Zeilen (bei Prozeduren)
- ClmDate - Datum der letzten Änderung
- ClmTime - Uhrzeit der letzten Änderung
- ClmType - Typ des Objekts
- ClmUserCreated - Erstellender Benutzer
- ClmUserModified - Benutzer der letzten Änderung
- ClmPerm - Effektiven Rechte des eigenen Benutzers

Column

 Spalteneintrag

Liste,
Eigenschaften
Spalte /
RecList,
Eigenschaften
Spalte /
Siehe RecView,
Eigenschaften
Spalte /
DataList,
Eigenschaften
Spalte /
StoList

Die Objekte RecList, RecView, DataList und StoList bestehen aus einem oder mehreren Spalteneinträgen.

Kontakt

StoListPopup



StoListPopup

Liste,

Siehe Eigenschaften,

Ereignisse

das Objekt StoListPopup entspricht dem Objekt StoList. Das Objekt wird lediglich in einem PopupList-Objekt angelegt.

Das Objekt besitzt zusätzlich die Eigenschaft ClmPopup, womit die Spalte definiert wird, deren Inhalt an das Eingabe-Objekt zurückgegeben wird.

RecList



RecList

[Liste](#),
[Eigenschaften](#),
[Ereignisse](#),
[Befehle](#),
[Fragen zur](#)
[RecList](#),

Siehe [Ereignisabläufe](#)

[RecList](#),
[Aktualisierung](#)
[der Inhalte](#)
[\(Blog\)](#),
[Editieren einer](#)
[Zelle \(Blog\)](#)

Mit diesem Objekt lassen sich Datensätze einer Datei oder Selektionsmenge in Form einer Liste anzeigen. In Abhängigkeit der darzustellenden Datensätze ist folgendes zu beachten:

- **Anzeige Datensätze einer Datei**

In der Eigenschaft [DbFileNo](#) wird die Dateinummer der gewünschten Datei angegeben. Über die Eigenschaft [DbKeyNo](#) wird bestimmt, über welchen Schlüssel die Liste sortiert werden soll.

- **Anzeige Datensätze einer verknüpften Datei**

Bei einer Verknüpfung wird die Zielfile in der Eigenschaft [DbLinkFileNo](#) und die Nummer der [Verknüpfung](#) in der Eigenschaft [DbKeyNo](#) angegeben.

- **Anzeige Datensätze einer Selektionsmenge**

Sollen Datensätze einer Selektionsmenge dargestellt werden, wird in der Eigenschaft [DbSelection](#) der Deskriptor der Selektion ([SelOpen\(\)](#)) angegeben. Da der Deskriptor der Selektion erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden.

- **Anzeige verknüpfter Datensätze einer Selektionsmenge**

Sollen verknüpfte Datensätze einer Selektionsmenge dargestellt werden, wird in der Eigenschaft [DbSelection](#) der Deskriptor der Selektion ([SelOpen\(\)](#)) angegeben. Da der Deskriptor der Selektion erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden. In der Selektionsmenge müssen die verknüpften Datensätze als Ergebnismenge enthalten sein.

- **Anzeige Datensätze über Filter**

Sollen Datensätze angezeigt werden, die einem bestimmten Filterkriterium entsprechen, wird in der Eigenschaft [DbFilter](#) der Deskriptor des gesetzten Filters ([RecFilterCreate\(\)](#)) angegeben. Da der Deskriptor des Filters erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden.

- **Anzeige von Grafiken**

Über die Eigenschaften [TileNameUser](#) und [ClmTypeImage](#) können Grafiken in

Kontakt

einer RecList angezeigt werden.

- Anzeige von Feldern aus einer anderen Datei

In dem RecList-Objekt können auch Spalten angegeben werden, die Felder aus einer anderen Datei ausgeben, als in der Eigenschaft DbFileNo angegeben ist. Der Inhalt der Felder muss in dem Ereignis EvtLstDataInit gesetzt werden. Auf die gleiche Weise können auch Abkürzungen in der Liste ausgeschrieben werden.

Die Feldpuffer werden durch die RecList nur dann mit dem selektierten Datensatz gefüllt, wenn beim Ereignis EvtLstSelect eine Funktion angesprungen wird. Der selektierte Datensatz wird erst beim Fokuswechsel in ein anderes Objekt in die Feldpuffer übertragen. Es werden nur die Feldpuffer gesetzt, die in den Spalten der RecList eingetragen sind. Das Übertragen der Feldpuffer beim Fokuswechsel kann durch das Deaktivieren des Flags WinLstRecFocusTermReset in der Eigenschaft LstFlags unterbunden werden.

Wird beim Ereignis EvtLstSelect eine Funktion aufgerufen, werden die Feldpuffer, die in den Spalten der RecList enthalten sind, mit den Werten des selektierten Datensatzes gefüllt. Diese Übertragung der Feldinhalte kann durch das Flag WinLstRecSelectBuffer in der Eigenschaft LstFlags aktiviert und deaktiviert werden.

Über das Ereignis EvtLstRecControl der RecList können bestimmte Sätze von der Anzeige ausgenommen werden.

Bei diesem Objekt ist eine Mehrfachselektion möglich. Zur Verwendung der Mehrfachselektion, muss die Eigenschaft MultiSelect gesetzt werden. Weitere Informationen befinden sich im Abschnitt SelectionData-Objekt.

Im Designer ist die Übernahme der Datenbankfelder in die RecList wie folgt möglich:

- Ziehen eines Feldes aus der Datenstruktur in die Liste.
- Ziehen eines Teildatensatzes aus der Datenstruktur in die Liste.
- Mit Betätigung der -Taste beim Ziehen wird automatisch eine RecList erstellt.
- Über den RecList-Editor.

Die Breite des RecList-Objekts ist durch Beschränkungen auf der Ebene des Betriebssystems auf ca. 16.000 Pixel begrenzt. Um ein flexibel verwendbares RecList-Objekt zu erstellen, kann entweder eine RecList mit vielen Spalten angelegt werden, wobei alle Spalten, die nicht benötigt werden, nicht angezeigt werden (Eigenschaft Visible = false), oder es werden die Spalten einer RecList an die darzustellenden Felder (Eigenschaft DbFieldName) angepasst.

Column

 Spalteneintrag

Liste,
Eigenschaften
Spalte /
RecList,
Eigenschaften
Spalte /
Siehe RecView,
Eigenschaften
Spalte /
DataList,
Eigenschaften
Spalte /
StoList

Die Objekte RecList, RecView, DataList und StoList bestehen aus einem oder mehreren Spalteneinträgen.

RecListPopup



RecListPopup

Liste,

Siehe Eigenschaften,

Ereignisse

Mit diesem Objekt wird das RecList-Objekt als Combobox dargestellt. In diesem Fall ist die Liste ein Unterobjekt des PopupList-Objekts. Es lassen sich Datensätze einer Datei oder Selektionsmenge in Form einer Liste anzeigen.

In Abhängigkeit der darzustellenden Datensätze ist folgendes zu beachten:

- Anzeige Datensätze einer Datei

In der Eigenschaft DbFileNo wird die Dateinummer der gewünschten Datei angegeben. Über die Eigenschaft DbKeyNo wird bestimmt, über welchen Schlüssel die Liste sortiert werden soll.

- Anzeige Datensätze einer verknüpften Datei

Bei einer Verknüpfung wird die Zielfile in der Eigenschaft DbLinkFileNo und die Nummer der Verknüpfung in der Eigenschaft DbKeyNo angegeben.

- Anzeige Datensätze einer Selektionsmenge

Sollen Datensätze einer Selektionsmenge dargestellt werden, wird in der Eigenschaft DbSelection der Deskriptor der Selektion (SelOpen()) angegeben. Da der Deskriptor der Selektion erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden.

- Anzeige verknüpfter Datensätze einer Selektionsmenge

Sollen verknüpfte Datensätze einer Selektionsmenge dargestellt werden, wird in der Eigenschaft DbSelection der Deskriptor der Selektion (SelOpen()) angegeben. Da der Deskriptor der Selektion erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden. In der Selektionsmenge müssen die verknüpften Datensätze als Ergebnismenge enthalten sein.

- Anzeige Datensätze über Filter

Sollen Datensätze angezeigt werden die einem bestimmten Filterkriterium entsprechen, wird in der Eigenschaft DbFilter der Deskriptor des gesetzten Filters (RecFilterCreate()) angegeben. Da der Deskriptor des Filters erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden.

- Anzeige von Grafiken

Über die Eigenschaften TileNameUser und ClmTypeImage können Grafiken in einem RecListPopup-Objekt angezeigt werden.

- Anzeige von Feldern aus einer anderen Datei

In dem RecListPopup-Objekt können auch Spalten angegeben werden, die Felder aus einer anderen Datei ausgeben, als in der Eigenschaft DbFileNo angegeben ist. Der Inhalt der Felder muss in dem Ereignis EvtLstDataInit gesetzt werden. Auf die gleiche Weise können auch Abkürzungen in der Liste ausgeschrieben werden.

Kontakt

Über das Ereignis EvtLstRecControl der RecListPopup können bestimmte Sätze von der Anzeige ausgenommen werden.

Neben der Maus kann die Liste über die Tastenkombination  +  oder per Prozedur über die Eigenschaft PopupOpen aufgeklappt werden.

Die Übernahme der Datenbankfelder in die Liste ist wie folgt möglich:

- Ziehen eines Feldes aus der Datenstruktur in die Liste.
- Ziehen eines Teildatensatzes aus der Datenstruktur in die Liste.
- Über den RecListPopup-Editor

RecView



RecView

[Liste](#),
[Eigenschaften](#),
[Ereignisse](#),
[Befehle, Aufbau](#)
[eines RecView](#),
[Ereignisabläufe](#)
[RecView](#),
[Grundlagen](#)
[\(Blog\)](#)

Siehe [Gleichbleibende](#)

[Zeilenhöhe](#)
[\(Blog\)](#),
[Editieren eines](#)
[Items \(Blog\)](#),
[Dynamische](#)
[Werte \(Blog\)](#),
[Verknüpfte](#)
[Datensätze](#)
[\(Blog\)](#)

Das RecView-Objekt eignet sich für die Darstellung von Inhalten aus der Datenbank mit variabler Zeilenhöhe. Beispielsweise ist die Anzeige von Bildern, Text- und RTF-Inhalten möglich. Die anzuzeigenden Inhalte (Elemente) bestimmen sich aus der Spaltendefinition, die im [RecView-Editor](#) konfiguriert wird.

Elemente können mit unbegrenzter Höhe ([ContentHeightMax](#) = 0) dargestellt werden oder auf eine Höhe begrenzt werden. Ist die Eigenschaft [ContentHeightMax](#) < 0, hat das Element eine feste Höhe, die dem Betrag der Eigenschaft entspricht. Enthält die Eigenschaft einen Wert > 0, dann besitzt das Element eine Maximalhöhe des angegebenen Wertes.

Die Höhe einer Zeile leitet sich aus der Höhe des größten Elementes einer Zeile ab.

In Abhängigkeit der darzustellenden Datensätze ist folgendes zu beachten:

- Anzeige Datensätze einer Datei

In der Eigenschaft [DbFileNo](#) wird die Dateinummer der gewünschten Datei angegeben. Über die Eigenschaft [DbKeyNo](#) wird bestimmt, über welchen Schlüssel die Liste sortiert werden soll.

- Anzeige Datensätze einer verknüpften Datei

Bei einer Verknüpfung wird die Zielfile in der Eigenschaft [DbLinkFileNo](#) und die Nummer der [Verknüpfung](#) in der Eigenschaft [DbKeyNo](#) angegeben.

- Anzeige Datensätze einer Selektionsmenge

Sollen Datensätze einer Selektionsmenge dargestellt werden, wird in der Eigenschaft [DbSelection](#) der Deskriptor der Selektion ([SelOpen\(\)](#)) angegeben. Da der Deskriptor der Selektion erst zur Laufzeit bekannt ist, kann diese

Kontakt

Eigenschaft nicht im Designer gesetzt werden.

- Anzeige verknüpfter Datensätze einer Selektionsmenge

Sollen verknüpfte Datensätze einer Selektionsmenge dargestellt werden, wird in der Eigenschaft DbSelection der Deskriptor der Selektion (SelOpen()) angegeben. Da der Deskriptor der Selektion erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden. In der Selektionsmenge müssen die verknüpften Datensätze als Ergebnismenge enthalten sein.

- Anzeige Datensätze über Filter

Sollen Datensätze angezeigt werden, die einem bestimmten Filterkriterium entsprechen, wird in der Eigenschaft DbFilter der Deskriptor des gesetzten Filters (RecFilterCreate()) angegeben. Da der Deskriptor des Filters erst zur Laufzeit bekannt ist, kann diese Eigenschaft nicht im Designer gesetzt werden.

- Anzeige von Text- und RTF-Inhalten

Wird die Eigenschaft ContentType auf den Wert WinContentTypeText bzw. WinContentTypeRTF gesetzt, können Text- und RTF-Inhalte aus Datenbankfeldern, internen Texten, binären Objekten und externen Dateien angezeigt werden. Die Quelle des Inhaltes wird über die Eigenschaft DbFieldName für Datenbankfelder, oder FileName für andere Quellen definiert.

- Anzeige von Bildern

Über die Eigenschaften ContentType mit dem Wert WinContentTypeImage können Bilder in einem RecView angezeigt werden. Der Pfad des Bildes wird in der Eigenschaft FileName definiert.

- Anzeige von Feldern aus einer anderen Datei

In dem RecView-Objekt können auch Spalten angegeben werden, die Felder aus einer anderen Datei ausgeben, als in der Eigenschaft DbFileNo angegeben ist. Die Caption...-Eigenschaft des jeweiligen (Sub-)Items muss im Ereignis EvtLstGroupInit auf den Inhalt des Felder der anderen Datei gesetzt werden. Auf die gleiche Weise können auch Abkürzungen in der Liste ausgeschrieben werden.

- Anzeige von beliebigen Werten

Zur Laufzeit kann im Ereignis EvtLstGroupInit die Eigenschaft Caption... mit dem jeweiligen Typ direkt gesetzt werden. Somit werden keine Hilfsfelder mehr benötigt.

Über das Ereignis EvtLstRecControl des RecViews können bestimmte Sätze von der Anzeige ausgenommen werden.

Im Designer ist die Übernahme der Datenbankfelder in das RecView wie folgt möglich:

- Ziehen eines Feldes aus der Datenstruktur in das View.
- Ziehen eines Teildatensatzes aus der Datenstruktur in das View.
- Mit Betätigung der - und der -Taste beim Ziehen wird automatisch ein RecView erstellt.
- Über den RecView-Editor.

Kontakt

Die Breite des RecView-Objekts ist durch Beschränkungen auf der Ebene des Betriebssystems auf ca. 16.000 Pixel begrenzt. Um ein flexibel verwendbares RecView-Objekt zu erstellen, kann entweder ein RecView mit vielen Spalten angelegt werden, wobei alle Spalten, die nicht benötigt werden, nicht angezeigt werden (Eigenschaft Visible = false), oder es werden die Spalten eines RecView an die darzustellenden Felder (Eigenschaft DbFieldName) angepasst.

Column

 Spalteneintrag

Liste,
Eigenschaften
Spalte /
RecList,
Eigenschaften
Spalte /
Siehe RecView,
Eigenschaften
Spalte /
DataList,
Eigenschaften
Spalte /
StoList

Die Objekte RecList, RecView, DataList und StoList bestehen aus einem oder mehreren Spalteneinträgen.

Kontakt

RecView-Aufbau

Aufbau eines RecView-Objektes

Ein RecView-Objekt kann aus folgenden Elementen bestehen:

- Column
- SubColumn
- View
- Group
- Item
- SubItem

Column

Ein RecView kann mehrere Spalten (Columns) enthalten. Diese können im RecView-Editor angelegt und deren Eigenschaften gesetzt werden.

SubColumn

Eine Spalte kann mehrere untergeordnete Spalten (SubColumns) enthalten. Sie haben die gleichen Eigenschaften wie die Spalten. Mit der Tastenkombination

[strg] + [↑] + [↓] kann eine Spalte in die Unterebene und mit [strg] + [↑] + [↑] in die Hauptebebene verschoben werden. SubColumns können keine Spalten untergeordnet werden. Über die Eigenschaften AreaLeft und AreaTop kann die Position der SubColumn innerhalb der Column definiert werden. Bei einer Column haben diese Eigenschaften keine Auswirkung.

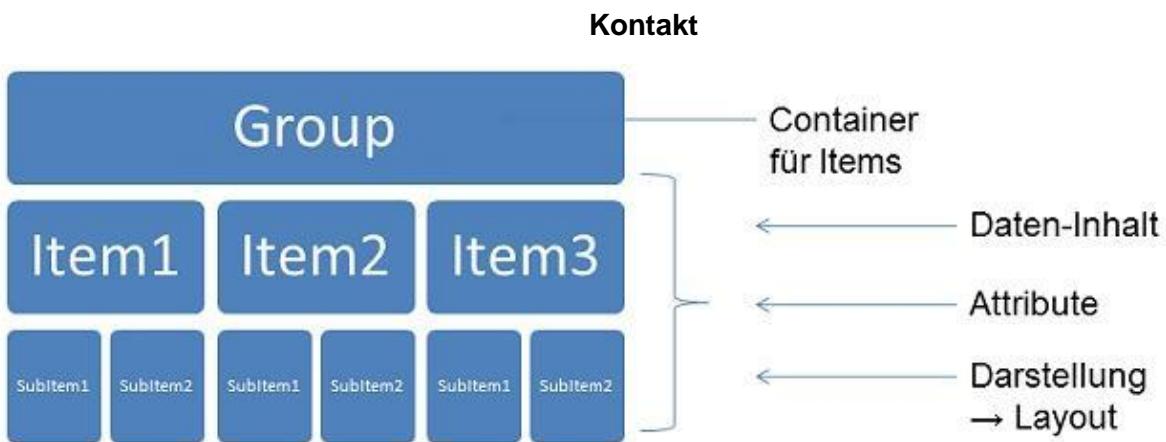
View

Ein RecView enthält 1 bis 4 Anzeigebereiche (Views), in denen die Datensätze dargestellt werden. In jedem View kann die Darstellung variiert werden.

Group

Zur Laufzeit wird für jeden angezeigten Datensatz eine Gruppe (Group) angelegt. Sie ist der Container, der die Items und SubItems enthält. Folgende Eigenschaften stehen bereit:

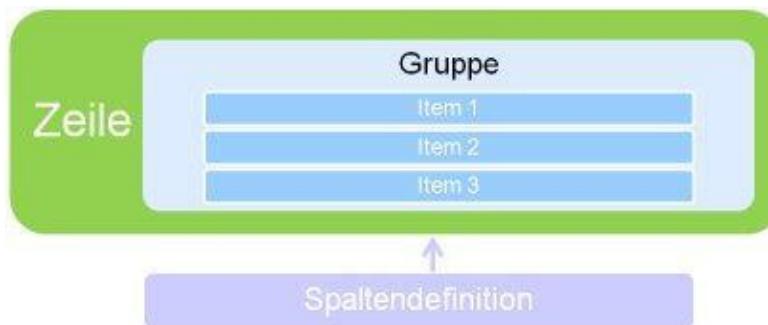
<u>ExpireTime</u>	Zeit in Sekunden, wann frühestens das nächste mal das Ereignis <u>EvtLstGroupInit</u> für die Gruppe aufgerufen wird.
<u>SelectorItem</u>	Item eines RecView-Objektes
<u>SelectorSubItem</u>	SubItem eines RecView-Objektes
<u>GroupColFg</u>	Vordergrundfarbe der Gruppe
<u>GroupColBkg</u>	Hintergrundfarbe der Gruppe
<u>GroupDrawSelect</u>	Selektionsrahmen zeichnen



Wurde ein Item oder SubItem durch setzen der Eigenschaften SelectorItem und SelectorSubItem gesetzt, können die Eigenschaften des jeweiligen Items bei der Gruppe verändert werden.

Item

Jede Column wird zur Laufzeit bei der Erstellung der Group als Item kopiert. Sie besitzt Eigenschaften die von gleichnamigen Eigenschaften der Spalte kopiert werden. Im Ereignis EvtLstGroupInit können die Eigenschaften des Items geändert werden. Zuvor muss das Item mit der Eigenschaft SelectorItem der Gruppe ausgewählt werden. Die Nummer unter der ein Item angesprochen wird, ist identisch mit der Nummer aus der Spaltendefinition im RecView-Editor. Diese Nummer ist unabhängig von der Anzeigereihenfolge.



SubItem

Jede SubColumn wird zur Laufzeit als SubItem kopiert. Um die Eigenschaften eines SubItems im Ereignis EvtLstGroupInit zu ändern, muss die Eigenschaft SelectorSubItem der Gruppe auf die Nummer des SubItems gesetzt werden.

TreeView



TreeView

Liste,

Siehe Eigenschaften,

Ereignisse,

Befehle

Mit diesem Objekt werden hierarchische Strukturen dargestellt. Das TreeView-Objekt besteht aus einem oder mehreren Knoten. Die Knoten werden mit dem TreeView-Editor erstellt und geändert.

Über die Eigenschaft NodeStyle kann aus einer Reihe von vordefinierten Symbolen ein Symbol für den jeweiligen Knoten ausgewählt werden. Das Verwenden eigener Symbole ist ebenfalls möglich. Dazu muss eine entsprechende TreeView-Grafik in die Datenbank importiert werden. In der Eigenschaft ImageTileUser wird dann die Nummer der zu verwendenden Kachel angegeben.

Das Anlegen und Entfernen der Knoten während der Laufzeit erfolgt über die Befehle WinTreeNodeAdd() und WinTreeNodeRemove(). Besitzt ein Knoten untergeordnete Knoten, werden die Symbole bzw. angezeigt. Das Symbol wird ebenfalls angezeigt, wenn bei dem Knoten die Eigenschaft NodeDynamic gesetzt ist.

Bei diesem Objekt ist eine Mehrfachselektion möglich. Zur Verwendung der Mehrfachselektion, muss die Eigenschaft MultiSelect gesetzt werden. Weitere Informationen befinden sich im Abschnitt SelectionData-Objekt.

Im TreeView kann eine Suche durchgeführt werden. Dazu muss die Eigenschaft SearchEnabled auf true gesetzt sein. Mit den, in den Eigenschaften SearchKeyStart, SearchKeyNext und SearchKeyPrev definieren, Tasten wird die Suche gestartet und fortgesetzt. Die Suchoptionen werden mit der Eigenschaft SearchFlags festgelegt. Standardmäßig wird mit einem Wildcard-Vergleich geprüft, ob die Eigenschaft Caption den Suchtext enthält. Die Groß-/Kleinschreibung wird dabei ignoriert. Die Suche kann mit der Funktion WinTreeNodeSearch() auch prozedural gestartet werden.

Kontakt

TreeNode



TreeNode

Siehe [Liste](#),

[Eigenschaften](#)

Das Objekt [TreeView](#) besteht aus mehreren Knoten. Das Anlegen und Entfernen der Knoten während der Laufzeit erfolgt über die Befehle [WinTreeNodeAdd\(\)](#) und [WinTreeNodeRemove\(\)](#).

Ausgabe-Objekte

Siehe Alle Oberflächen-Objekte

Label



Icon



Picture



MetaPicture



Animation



DocView



WebNavigator



PrtJobPreview



Barcode



Chromium



Kontakt

Label

 Bezeichner

Liste,

Siehe Eigenschaften,

Ereignisse

Mit einem Label kann ein kurzer, in der Regel statischer Text ausgegeben werden. Der Text des Labels wird in der Eigenschaft Caption gesetzt werden. Das Zeichen & wird verwendet, um einen "Hot-Spot" zu definieren. Der nachfolgende Buchstabe wird unterstrichen. Ist in der Eigenschaft ObjLink ein Eingabe-Objekt eingetragen, wird

beim Drücken der Tastenkombination  +<unterstrichener Buchstabe> der Fokus in das Eingabe-Objekt versetzt. Um das Zeichen & auszugeben, muss es zwei mal geschrieben werden.

In der Eigenschaft DbFieldName kann ein Datenbankfeld von beliebigem Typ mit dem Eingabeobjekt verknüpft werden. Der Inhalt des Feldes wird dann anstelle der Caption angezeigt. Eine Auswertung des & findet dann nicht statt.

Kontakt

Icon



Symbol

Liste,

Siehe Eigenschaften,
Ereignisse

Über das Objekt Icon lassen sich Piktogramme vom Typ ICO anzeigen. Die ICO-Datei kann dabei extern oder in der Datenbank vorliegen.

In der Eigenschaft FileName wird der Name der darzustellenden Datei angegeben. Wird eine Bilddatei bestehend aus mehreren Icons verwendet, ist in der Eigenschaft NumIcon die Bildnummer der Icon-Grafik anzugeben. Wird die Eigenschaft auf -1 gesetzt, wird die Eigenschaft TileSize verwendet, um die Größe zu definieren. Handelt es sich bei dem Symbol um eine externe Datei muss in der Eigenschaft TypeIcon die Einstellung WinIcoExternFile verwendet werden.

Über diese Eigenschaft kann auch aus einer Reihe von vorgegebenen Standard-Symbolen (Warnsymbol, Fehlersymbol ...) gewählt werden.

Picture



Objekt zum Darstellen von Grafiken

[Liste](#),
[Eigenschaften](#),
[Ereignisse](#),
[PrtPicture](#),
[Darstellung](#)

Siehe von

[Kennzahlen](#)
[\(Blog\)](#),
[Grafiken in CONZEPT 16](#)
[\(Blog\)](#)

In diesem Objekt können über die Eigenschaft Caption Grafiken vom Format BMP, JPG, GIF, PNG, TIFF (ein und mehrseitig) und ICO dargestellt werden. Die anzuzeigende Grafik kann extern oder in der Datenbank vorliegen. Die Angabe einer externen Datei kann mit einem absoluten oder relativen Pfad erfolgen. Bei der Angabe eines absoluten Pfades wird ein '*', bei der eines relativen Pfades (relativ zum Startverzeichnis des CONZEPT 16-Clients) wird ein '!' vorangestellt. Bei mehrseitigen TIFF- und ICO-Bildern, wird die anzuzeigende Seite in der Eigenschaft CurrentInt angegeben.

In einem Picture-Objekt können weitere Picture- oder MetaPicture-Objekte eingefügt werden (Overlays). Dies erfolgt im Designer, indem ein Objekt selektiert wird und anschliessend über die Palette ein entsprechendes Objekt eingefügt wird. Untergeordneten Objekten können keine weiteren Objekte hinzugefügt werden.

Durch Setzen der Eigenschaft ColBkg eines untergeordneten Objekts auf WinColTransparent wird der Inhalt des übergeordneten Objekts im untergeordneten Objekt gezeichnet. Die Eigenschaft Opacity des untergeordneten Objekts bestimmt in diesem Fall die Deckfähigkeit zum Hintergrund (dem Bild des übergeordneten Bildes).

Auf diese Weise können z. B. einfach Stempel oder Wasserzeichen für die Anzeige eines Bildes generiert werden.

Ist die darzustellende Grafik in der Datenbank als binäres Objekt gespeichert, kann es durch die Angabe von '>0' gefolgt vom Pfad und dem Namen des Objekts in der Eigenschaft Caption werden. Ist das binäre Objekt verschlüsselt gespeichert, muss der Schlüssel in der Eigenschaft CryptKey angegeben werden. Bei mit DbaConnect() verbundenen Datenbanken wird nach '>' der entsprechende Dateinummernbereich (z. B. '>2') angegeben.



Bei Angabe des Namens ohne Präfix ('!' und '*') wird eine Bilddatei in der Datenbank referenziert. Über die Ressourcenverwaltung lassen sich Grafiken der unterstützten Formate in die Datenbank einlesen.

Beispiele:

```
'!Picture\Artikel.tif'          // externe Datei relativ angegeben'*C:\C16\Picture\Artikel.tif'
```

Wird ein Bild dargestellt, dass größer als das Objekt ist, erscheinen automatisch

Kontakt

Scrollbalken auf der rechten und der unteren Seite des Objektes (die Rollbalken können mit der Eigenschaft ScrollbarVisible deaktiviert werden). Bei Angabe von 0 bei der Eigenschaft ZoomFactor wird das Bild so skaliert, dass es komplett im Picture-Objekt dargestellt wird.

Neben der üblichen Funktionalität mit der linken Maustaste, kann das Bild mit dem Rad einer Wheelmaus gescrollt werden: Durch Drehen des Rades wird vertikal gescrollt. Erfolgt das Drehen bei gedrückter -Taste wird horizontal gescrollt. Das Scrolling erfolgt pixelweise. Um ein schnelleres Scrolling zu erreichen, muss zusätzlich die -Taste gedrückt gehalten werden.

Zusätzlich werden noch folgende Tastenkommandos akzeptiert:

Taste	Wirkung
	Bild eine Stufe heranzoomen
	Bild eine Stufe herauszoomen
+	Höchste Zoomstufe
+	niedrigste Zoomstufe
	Bild nach oben scrollen
	Bild nach unten scrollen
	Bild nach links scrollen
	Bild nach rechts scrollen
	Bild an den Anfang scrollen
	Bild an das Ende scrollen
	Bild schneller nach oben scrollen
	Bild schneller nach unten scrollen

Die Taste ändert in Kombination mit den Pfeiltasten die Geschwindigkeit des scrollens. Bei den Tasten , , und kann die Richtung der Scrollbewegung durch Drücken der Taste umgekehrt werden.

- Ist das Ereignis EvtKeyItem beim Picture-Objekt eingetragen, entscheidet der Rückgabewert darüber, ob die Scrolltasten verarbeitet werden (true) oder nicht (false). Die Zoomfunktion der / Tasten wird immer durchgeführt.

MetaPicture



Objekt zur Darstellung von Vektorgrafiken

[Liste](#),

[Eigenschaften](#),

Siehe [Ereignisse](#),

[Grafiken in](#)

[CONZEPT 16](#)

[\(Blog\)](#)

In diesem Objekt können über die Eigenschaft Caption Vektorgrafiken im Metafile-Format (EMF, WMF) dargestellt werden. Die anzuzeigende Grafik kann extern oder in der Datenbank vorliegen. Die Angabe einer externen Datei kann mit einem absoluten oder relativen Pfad erfolgen. Bei der Angabe eines absoluten Pfades wird ein '*', bei der eines relativen Pfades (relativ zum Startverzeichnis des CONZEPT 16-Clients) wird ein '!' vorangestellt.

In einem MetaPicture-Objekt können weitere Picture- oder MetaPicture-Objekte eingefügt werden (Overlays). Dies erfolgt im Designer, indem ein Objekt selektiert wird und anschliessend über die Palette ein entsprechendes Objekt eingefügt wird. Untergeordneten Objekten können keine weiteren Objekte hinzugefügt werden.

Durch Setzen der Eigenschaft ColBkg eines untergeordneten Objekts auf

WinColTransparent wird der Inhalt des übergeordneten Objekts im untergeordneten Objekt gezeichnet. Die Eigenschaft Opacity des untergeordneten Objekts bestimmt in diesem Fall die Deckfähigkeit zum Hintergrund (dem Bild des übergeordneten Bildes).

Auf diese Weise können z. B. einfach Stempel oder Wasserzeichen für die Anzeige eines Bildes generiert werden.

Ist die darzustellende Grafik in der Datenbank als binäres Objekt gespeichert, kann es durch die Angabe von '>0' gefolgt vom Pfad und dem Namen des Objekts in der Eigenschaft Caption werden. Ist das binäre Objekt verschlüsselt gespeichert, muss der Schlüssel in der Eigenschaft CryptKey angegeben werden. Bei mit DbaConnect() verbundenen Datenbanken wird nach '>' der entsprechende Dateinummernbereich (z. B. '>2') angegeben.



Bei Angabe des Namens ohne Präfix ('!' und '*') wird eine Bilddatei in der Datenbank referenziert. Über die Ressourcenverwaltung lassen sich Grafiken der unterstützten Formate in die Datenbank einlesen.

Beispiele:

```
'!Picture\Artikel.emf'          // externe Datei relativ angegeben'*C:\C16\Picture\Artikel.emf'
```

Wird ein Bild dargestellt, dass größer als das Objekt ist, erscheinen automatisch Rollbalken auf der rechten und der unteren Seite des Objektes (die Rollbalken können mit der Eigenschaft ScrollbarVisible deaktiviert werden). Bei Angabe von 0 bei der Eigenschaft ZoomFactor wird das Bild so skaliert, dass es komplett im MetaPicture-Objekt dargestellt wird.

Neben der üblichen Funktionalität mit der linken Maustaste, kann das Bild mit dem Rad einer Wheelmaus gescrollt werden: Durch Drehen des Rades wird vertikal

Kontakt

gescrollt. Erfolgt das Drehen bei gedrückter -Taste wird horizontal gescrollt. Das Scrolling erfolgt pixelweise. Um ein schnelleres Scrolling zu erreichen, muss zusätzlich die -Taste gedrückt gehalten werden.

Zusätzlich werden noch folgende Tastenkommandos akzeptiert:

Taste	Wirkung
	Bild eine Stufe heranzoomen
	Bild eine Stufe herauszoomen
+	Höchste Zoomstufe
+	niedrigste Zoomstufe
	Bild nach oben scrollen
	Bild nach unten scrollen
	Bild nach links scrollen
	Bild nach rechts scrollen
	Bild an den Anfang scrollen
	Bild an das Ende scrollen
	Bild schneller nach oben scrollen
	Bild schneller nach unten scrollen

Die Taste ändert in Kombination mit den Pfeiltasten die Geschwindigkeit des scrollens. Bei den Tasten , , und kann die Richtung der Scrollbewegung durch Drücken der Taste umgekehrt werden.



Ist das Ereignis EvtKeyItem beim MetaPicture-Objekt eingetragen, entscheidet der Rückgabewert darüber, ob die Scrolltasten verarbeitet werden (true) oder nicht (false). Die Zoomfunktion der / Tasten wird immer durchgeführt.

Animation



.AVI Datei-Objekt
[Liste](#),

Siehe [Eigenschaften](#),
[Ereignisse](#)

In diesem Objekt können AVI-Dateien (Audio Video Interleave) abgespielt werden. Der Film ist in der Eigenschaft [FileName](#) anzugeben. Über die Eigenschaft [TypeFile](#) steht eine Liste verschiedener Animationen des Betriebssystems zur Verfügung.

Kontakt

DocView



Anzeige-Objekt für Bilder und PDF-Dokumente

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#),

[PrtPdf](#)

Das DocView-Objekt ermöglicht die Anzeige von PDF-Dokumenten ohne dass hierzu entsprechende Software eines anderen Anbieters auf dem Rechner installiert sein muss. Neben der Anzeige von PDF ist auch die Anzeige von TIFF, JPEG, PNG, BMP und GIF möglich.

Der Pfad und der Name des Dokuments wird in der Eigenschaft [FileName](#) angegeben.

Das DocView-Objekt erlaubt eine Galerie-Anzeige mehrerer PDF- bzw. TIFF-Seiten. Es werden dabei mehrere Seiten dargestellt. Die Anzahl der Seiten, die nebeneinander dargestellt werden, wird über die Eigenschaft [PageNumClm](#) definiert.

Die Hintergrundfarbe der Seiten kann über die Eigenschaft [ColBkg](#) gesetzt werden; die Farbe des Seitenrandes über die Eigenschaft [ColBoundary](#).

Die Eigenschaft [PageZoom](#) gibt an, wie die Seite oder die Seiten dargestellt werden:

[PrtPageZoomUser](#)

Benutzerdefinierter Zoom

[PrtPageZoomPage](#)

Zoom auf ganze Seite

[PrtPageZoomPageWidth](#)

Zoom auf Seitenbreite

[PrtPageZoomPageAll](#)

Zoom auf alle Seiten

[PrtPageZoomPageWidthAll](#) Zoom auf Seitenbreite aller Seiten

Innerhalb des Objekts können zusätzliche [Picture](#)- oder [MetaPicture](#)-Objekte eingefügt werden (Overlay). Dies erfolgt im Designer, indem das DocView-Objekt selektiert wird und anschließend über die [Palette](#) ein Objekt eingefügt wird. In den Eigenschaften [ViewId](#) und [ViewType](#) kann angegeben werden, unter welchen Bedingungen das Overlay angezeigt wird.

WebNavigator



Anzeige registrierter Dateitypen

[Liste](#),

[Eigenschaften](#),

[Ereignisse](#),

Siehe [Kompatibilität](#)

[\(Blog\)](#),

[Steuerung](#)

[\(Blog\)](#)

In diesem Objekt können beliebige registrierte Dateitypen angezeigt werden. Die Datei wird innerhalb des WebNavigators angezeigt und kann je nach verwendetem Programm auch geändert werden.

In der Eigenschaft [Caption](#) wird der Dateiname der anzuzeigenden Datei oder eine Web-Adresse (URL) angegeben.

Ist bei der Initialisierung des Objekts ein Fehler aufgetreten, kann dieser mit der Anweisung [ComInfo\(\)](#) ermittelt werden.

Mit dem Befehl [WinWbnExecCommand\(\)](#) können Kommandos für die Navigation und die Zwischenablage an den WebNavigator gesendet werden.

Kontakt

PrtJobPreview

PrtJobPreview

Liste,

Siehe Eigenschaften,

Ereignisse

In diesem Objekt können über die Eigenschaft Caption Druckjob-Dateien und in der Datenbank gespeicherte PrintDoc-Objekte angezeigt werden. Der Druckjob muss zu diesem Zeitpunkt geschlossen (PrtJobClose()) sein.

Die Angabe einer extern vorliegenden Druckjob-Datei kann mit einem absoluten oder relativen Pfad erfolgen. Bei der Angabe eines absoluten Pades wird ein '*', bei der eines relativen Pfades (relativ zum Startverzeichnis des CONZEPT 16-Clients) wird ein '!' vorangestellt. Liegt die Datei als binäres Objekt vor, können diese mit dem Präfix '>' angesprochen werden (siehe Caption).

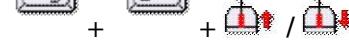
Bei Angabe des Namens ohne Präfix ('!', '*' und '>') wird ein PrintDoc-Objekt aus der Datenbank geladen.

Mit der Eigenschaft CryptKey kann ein Passwort gesetzt werden, welches benötigt wird, wenn ein verschlüsselter Druckjob gelesen werden soll.

Die Eigenschaft CurrentInt definiert die Nummer der anzuzeigenden Seite. MaxInt kann nur gelesen werden und liefert die Anzahl der Seiten. Diese Eigenschaften können nur zur Laufzeit gesetzt werden.

-  Wird in dem Druckjob ein PrtRtf-Objekt ausgegeben, kann es in der Vorschau zu Abweichungen gegenüber dem Druckergebnis kommen, da noch nicht bekannt ist, über welchen Drucker der Ausdruck erfolgt. Ist zum Zeitpunkt der Anzeige des Druckjobs bereits bekannt, auf welchem Drucker das Dokument gedruckt werden soll, können die Abweichungen zwischen Vorschau und Druckergebnis minimiert werden, in dem der Deskriptor des PrintDevice-Objekts der Eigenschaft PrtDevice zugewiesen wird.

Bei Verwendung einer Wheel-Maus werden folgende zusätzliche Funktionen unterstützt:

- Vertikales Scrollen - Bildlaufrad 
- Horizontales Scrollen -  + 
- Vertikales Scrollen mit größerer Schrittweite - 
- Horizontales Scrollen mit größerer Schrittweite - 

In dem PrtJobPreview-Objekt kann die Anzahl der Vorschauseiten variiert werden. Durch Angabe der Eigenschaft PageCount kann definiert werden, wie viele Seiten ab der, durch die Eigenschaft CurrentInt gesetzten, Startseite angezeigt werden sollen. Durch die Eigenschaft PageNumClm kann angegeben werden, wie viele Seiten maximal nebeneinander dargestellt werden.

Barcode



[Barcode anzeigen](#)

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#),

[Befehle](#)

Dieses Objekt stellt Barcodes oder Strichcodes dar. Der Barcode kann über die Eigenschaften [Caption](#) oder [DbFieldName](#) gesetzt werden. Der Typ des Barcodes wird in der Eigenschaft [TypeBarcode](#) oder als Präfix angegeben. Die Liste der unterstützten Barcodes befindet sich in der Eigenschaft [TypeBarcode](#).

Das Format wird durch die Länge der Barcodenummer und der integrierten Prüfziffern automatisch erkannt. Um ein Format zu erzwingen, kann der Barcodenummer ein Präfix vorangestellt werden: 'EAN', 'UPC', 'ISBN', 'I25', 'Code39N', 'Code39C', 'Code128B', 'Code128C' oder 'Code128X'. Der Typ kann ebenfalls in der Eigenschaft [TypeBarcode](#) angegeben werden.

Beispiele:

```
'3883224537' 'ISBN3-88322-453-7' 'Code39NABC123'
```

Kontakt

Chromium



Web-Browser-Objekt
[Liste](#), [Eigenschaften](#),

Siehe [Ereignisse](#),
[CodeLibrary-Beispiel](#)
"ChromeBrowser"

Das Chromium-Objekt ermöglicht die Anzeige Web-basierter Inhalte.

Das Objekt stellt eine moderne Alternative zum [WebNavigator](#)-Objekt dar und basiert auf dem quelloffenen [Chromium-Projekt](#) auf dem auch der [Browser Chrome](#) basiert. Das Chromium-Projekt ist über das [Chromium Embedded Framework](#) in CONZEPT 16 integriert.

Das Objekt ist optional und kann über die Setup-Routine als eigenständiges Paket installiert werden.

Die Anzeige einer Web-Seite geschieht durch setzen der Eigenschaft [Caption](#). Mit dem Befehl WinCroNavigate kann zu einer zuvor angezeigten Seite navigiert werden. Mit WinCroReload kann die aktuelle Seite erneut geladen werden.

Die Entwickler-Werkzeuge können mit den Eigenschaften [CroDevToolsSide](#) und [CroDevToolsSize](#) aktiviert werden.

Kontakt

Eigenschaften eines Chromium-Objekts

Das Objekt Chromium unterstützt folgende Eigenschaften.

Chromium,

Liste der

Siehe Objekte, Liste

der

Eigenschaften

- AlignGrouping
- AlignHeight
- AlignMarginBottom
- AlignMarginLeft
- AlignMarginRight
- AlignMarginTop
- AlignWidth
- AreaBottom
- AreaHeight
- AreaLeft
- AreaRight
- AreaTop
- AreaWidth
- Caption
- ColDesign
- CroDevToolsSide
- CroDevToolsSize
- Custom
- Design
- DesignFlags
- Disabled
- Name
- StyleBorder
- TabPos
- Url
- Visible
- ZoomFactor

Kontakt

Ereignisse eines Chromium-Objekts

Das Objekt Chromium unterstützt folgende Ereignisse.

Chromium,

Liste der

Siehe Objekte,

Liste der

Ereignisse

- EvtChanged
- EvtChangedDesign
- EvtCroNavigate
- EvtKeyItem
- EvtMouse

Toolbar-Objekte

Siehe Alle Oberflächen-Objekte

FrameClient



ToolbarDock



Toolbar



Toolbar für Copy Paste



Toolbar für MDI



ToolbarRtf

ToolbarMenu



Windowbar

Statusbar

Kontakt

FrameClient



FrameClient

Liste,

Siehe Eigenschaften,
Ereignisse

Das FrameClient-Objekt kann nur in ein Frame- oder MdiFrame-Objekt eingefügt werden.

Es dient dazu, das Frame-Objekt so aufzuteilen, dass ein oder mehrere Toolbar-Objekte im Frame neben den anderen im Frame schon vorhandenen Objekten Platz haben.

Wird ein FrameClient-Objekt einem Frame-Objekt hinzugefügt und besitzt das Frame-Objekt bereits untergeordnete Objekte, so werden alle Unterobjekte in das FrameClient-Objekt eingefügt. Nachfolgende Operationen (Drag & Drop, Copy & Paste, Erstellen von Objekten) geschieht dann im FrameClient, da dieser von nun an die Unterobjekte enthält.

In ein AppFrame-Objekt kann kein FrameClient-Objekt eingefügt werden und ist dort auch nicht notwendig.

Kontakt

ToolbarDock



ToolbarDock

Liste,

Siehe Eigenschaften,

Ereignisse

Das ToolbarDock-Objekt dient als Container für Toolbar-Objekte. Ein ToolbarDock-Objekt kann maximal fünf Toolbar-Objekte aufnehmen. Ein Toolbar-Objekt kann nicht in ein ToolbarDock-Objekt eingefügt werden, wenn dieses bereits ein Statusbar-Objekt enthält.

Umgekehrt ist es auch nicht möglich ein Statusbar-Objekt in ein ToolbarDock-Objekt einzufügen, das bereits ein oder mehrere Toolbars enthält.

Das ToolbarDock-Objekt kann in alle Frame-Objekte (Frame, MdiFrame und AppFrame) eingesetzt werden.

Durch die Eigenschaft DockSide kann bestimmt werden, an welcher Seite des Fensters das ToolbarDock-Objekt dargestellt werden soll. Um im unteren Bereich des Fensters sowohl eine Statuszeile, als auch eine Toolbar anzuzeigen, müssen zwei ToolbarDock-Objekte erzeugt und die Eigenschaft DockSide einmal auf WinDockBottom und einmal auf WinDockBottomTop gesetzt werden.

Bei horizontalen ToolbarDock-Objekten, kann die Höhe, bei vertikalen ToolbarDock-Objekten die Breite beliebig verändert werden. Zur Laufzeit passt sich die Höhe bzw. die Breite des Objektes an die beinhalteten Objekte an.

Toolbar

 **Toolbar**

Liste,

Siehe Eigenschaften,

Ereignisse

Bei dem Toolbar-Objekt handelt es sich um einen Symbolbalken mit Buttons, die auf Mausklick oder eine Funktionstaste zugeordnete Funktion auslösen. Bevor das Toolbar-Objekt angelegt werden kann, muss das Objekt ToolbarDock in dem Frame eingebunden werden. Eine Toolbar kann am oberen, unteren, rechten und linken Rand eines Fensters dargestellt werden.

Die Buttons einer Toolbar werden mit dem Toolbar-Editor bearbeitet. Durch Doppelklick auf die Toolbar oder Auswahl des Menüpunktes "Toolbar bearbeiten" im Kontextmenü wird dieser erreicht.

Über die Palette können bereits mit Schaltflächen versehene Toolbars eingefügt werden.

Alle Schaltflächen sind vom Typ Toolbar-Button mit den entsprechenden Eigenschaften und Ereignissen.

Die Toolbar für das RTF-Objekt ist ein separates Objekt, das nicht über den Editor für die Toolbar bearbeitet werden kann.

Kontakt

Toolbar-Button



Toolbar-Button

Siehe [Liste](#),

Eigenschaften

Eine [Toolbar](#) besteht aus einem oder mehreren Toolbar-Buttons. Das Auswählen eines Buttons kann mit der Maus oder über eine Funktionstaste erfolgen.

Für die Auswahl über Funktionstaste wird in der Eigenschaft [MenuKey](#) die gewünschte Funktionstaste angegeben. Neben einer Beschriftung ([ImageOption](#)) kann ein Button auch über ein Symbol verfügen. In der Eigenschaft [ImageTile](#) kann aus einer Reihe von vordefinierten Symbolen gewählt werden.

Das Verwenden eigener Symbole ist ebenfalls möglich. Dazu muss eine entsprechende Grafik in die Datenbank [importiert](#) und den Eigenschaften [TileNameNormal](#), [TileNamePressed](#) und [TileNameSelected](#) zugewiesen werden. In der Eigenschaft [ImageTileUser](#) wird dann die Nummer der zu verwendenden Kachel angegeben.

Ist die Werkzeugleiste zu klein, um alle Schaltflächen anzeigen zu können, wird am rechten bzw. unteren Rand der Werkzeugleiste die ">>" Schaltfläche angezeigt. Wird diese betätigt, erscheint ein Popup-Menü, sofern bei den nicht sichtbaren Schaltflächen die Eigenschaft [Caption](#) einen Wert enthält. Zusätzlich wird links neben dem Text ein Bild (Eigenschaft [ImageTile](#)) angezeigt, sofern vorhanden. Ist die Eigenschaft [TileNameMenu](#) gesetzt, werden nicht die Bilder des Toolbar-Objektes sondern die Bilder der hier angegebenen Grafik verwendet. Die Nummer der Kachel ergibt sich weiterhin aus der Eigenschaft [ImageTile](#).

Buttons können nur im [Toolbar-Editor](#) erstellt werden.

Durch Anklicken eines Toolbar-Buttons (oder durch Eingeben des Tastatur-Kürzels) wird das Ereignis [EvtMenuCommand](#) des Fenster-Objektes ausgelöst. Hier kann der ausgewählte Button ermittelt und die entsprechende Funktionalität aufgerufen werden. Das Ereignis [EvtMenuCommand](#) beim Toolbar-Button wird nur ausgelöst, wenn ein Eintrag aus dem Kontextmenü des Buttons ausgewählt wird.

Toolbar für Copy Paste



Toolbar für Copy Paste

Liste,

Siehe Eigenschaften,

Ereignisse,

Toolbar

Diese Toolbar enthält folgende Schaltflächen:

Name	Darstellung
------	-------------

tbnEditCut	
------------	--

tbnEditCopy	
-------------	--

tbnEditPaste	
--------------	--

tbnEditDelete	
---------------	--

tbnEditUndo	
-------------	--

tbnEditRedo	
-------------	--

Toolbar für MDI

 **Toolbar für MDI**

Liste,

Siehe Eigenschaften,

Ereignisse,

Toolbar

Diese Toolbar enthält folgende Schaltflächen:

Name	Darstellung
------	-------------

tbnMdiCascade	
---------------	---

tbnMdiArrange	
---------------	--

tbnMdiTileVert	
----------------	--

tbnMdiTileHorz	
----------------	--

tbnMdiNext	
------------	--

tbnMdiPrev	
------------	---

Kontakt

ToolbarRtf

ToolbarRtf

Liste,
Eigenschaften,

Siehe Ereignisse,
Toolbar

Dieses Toolbar-Objekt ist speziell zur Verwendung mit dem RtfEdit-Objekt vorbereitet. Wird in ein Frame- oder MdiFrame-Objekt, in dem bereits ein RtfEdit-Objekt existiert, diese Toolbar eingefügt, verbindet sie sich automatisch mit dem bereits existierenden Objekt. Für die Verbindung wird die Eigenschaft ObjLink verwendet.

Bevor das Toolbar-Objekt angelegt werden kann, muss das Objekt ToolbarDock in dem Frame eingebunden werden. Eine Toolbar kann am oberen, unteren, rechten und linken Rand eines Fensters dargestellt werden.

Die Toolbar stellt alle notwendigen Funktionen zum Editieren und Formatieren des Textes zur Verfügung.

In einem ToolbarRtf-Objekt sind folgende Objekte integriert:

Name	Typ	Darstellung
edFontName	<u>FontNameEdit</u>	Arial
edFontSize	<u>FontSizeEdit</u>	10
btnRtfBold	<u>Button</u>	B
btnRtfItalic	<u>Button</u>	I
btnRtfUnderline	<u>Button</u>	U
btnRtfAlignLeft	<u>Button</u>	
btnRtfAlignCenter	<u>Button</u>	
btnRtfAlignRight	<u>Button</u>	
btnRtfAlignJustify	<u>Button</u>	
btnRtfColFg	<u>ColorButton</u>	
btnRtfColBkg	<u>ColorButton</u>	

Nicht benötigte Funktionen können aus der Toolbar über das Kontextmenü des Objektes oder mit der -Taste entfernt werden. Bei Objekten vom Typ Button kann die Funktion über den Menüeintrag RTF-Toolbar des Kontextmenüs geändert werden.

In die Toolbar können weitere Objekte beliebigen Typs aufgenommen werden.

ToolbarMenu

ToolbarMenu

Liste,

Siehe Eigenschaften,
Ereignisse

Im Modern Theme Style kann das klassische (von Windows generierte) Menü nicht mehr angezeigt werden. Eine an das klassische Menü angelehnte Interaktion und Darstellungsweise bietet hier das ToolbarMenu-Objekt. Wird es in ein ToolbarDock-Objekt eingefügt, wird das vorhandene Menü ausgeblendet und das ToolbarMenu-Objekt übernimmt die Menüfunktion.

Das Objekt besteht aus einer Toolbar, welche die Menüpunkte des im Frame eingesetzten Menüs (Eigenschaft MenuName) besitzt. Wird das Menü geändert, indem z. B. durch WinMenuItemAdd() einer neuen Menüpunkt hinzugefügt oder die Eigenschaft MenuName neu gesetzt wird, dann aktualisiert sich das ToolbarMenu-Objekt entsprechend.

Kompatibilität

Damit die Kompatibilität mit dem klassischen Menü gewährleistet ist, löst das ToolbarMenu-Objekt dieselben Ereignisse aus, wie das Menu-Objekt. Somit kann in den meisten Fällen, das Menü ohne programmtechnische Änderungen durch das ToolbarMenu-Objekt ersetzt werden. Ein Frame kann maximal ein ToolbarMenu-Objekt enthalten.



Die Schaltflächen des ToolbarMenu-Objektes repräsentieren die oberste Ebene des referenzierten Menüs. Jede Schaltfläche ist ein ToolbarButton-Objekt des entsprechenden Menüpunktes und erhält auch dieselbe Name-Eigenschaft. Um die Kompatibilität mit vorhandem Code zu gewährleisten, liefert WinSearch() und der \$-Operator / \$:-Operator jedoch das entsprechende MenuItem und nicht den ToolbarButton.

Wird das ToolbarMenu-Objekt in einem AppFrame verwendet, dann werden die für MDI typischen Schaltflächen (rechts vom Menü) über das ToolbarMenu-Objekt emuliert.

Kontakt

Windowbar

Windowbar

Liste,

Siehe Eigenschaften,

Ereignisse

Das Windowbar-Objekt kann nur in ein ToolbarDock-Objekt eingefügt werden. Es dient als Container-Objekt für beliebige andere Objekte. Es können alle Objekte, die in ein Frame-Objekt integriert werden können, ebenfalls in ein Windowbar-Objekt eingefügt werden.

Mit diesem Objekt wird es möglich innerhalb einer Werkzeugleiste zum Beispiel eine ComboBox oder ein TreeView zur Navigation innerhalb der Applikation zu integrieren.

Kontakt

Statusbar

Statusbar

Liste,

Siehe Eigenschaften,

Ereignisse

Die Statusbar dient der Anzeige von Informationen und kann nur an der unteren Seite eines Frame-Objekts dargestellt werden. Die Statusbar besteht aus einem oder mehreren Statusbar-Buttons.

Zum Bearbeiten einer Statusbar steht der Statusbar-Editor zur Verfügung.

Kontakt

Statusbar-Button



Statusbar-Button

Siehe [Liste](#),

Eigenschaften

Eine Statusbar besteht aus einem oder mehreren Statusbar-Buttons. In der Eigenschaft Caption können folgende Makrodefinitionen angegeben werden:

```
$ (DATE [, <datums-format>]) // Datum  
$ (TIME [, <zeit-format>]) // Zeit
```

Zur Laufzeit wird das lokale Datum, bzw. die lokale Uhrzeit ausgegeben. Die Aktualisierung erfolgt automatisch. Die Format-Zeichenkette <datums-format> und <zeit-format> sind optional und identisch zu denen der Eigenschaft ScalaLabels.

```
$ (KEY,NUM) // Anzeige des Toggle-Status der NumLock-Taste  
$ (KEY,INS) // Anzeige des Toggle-Status der Einfg.-Taste  
$ (KEY,CAP) // Anzeige des Toggle-Status der Umschalt-Taste
```

Der Status der Taste wird nur dann in der Statusbar angezeigt, wenn das Objekt das den Fokus besitzt ein Eingabeobjekt ist.

Neben einer Beschriftung (ImageOption) kann ein Button auch über ein Symbol verfügen. In der Eigenschaft ImageTile kann aus einer Reihe von vordefinierten Symbolen gewählt werden.

Das Verwenden eigener Symbole ist ebenfalls möglich. Dazu muss eine entsprechende Grafik in die Datenbank importiert werden. In der Eigenschaft ImageTileUser wird dann die Nummer der zu verwendenden Kachel angegeben.

Buttons können nur im Statusbar-Editor erstellt werden.

Anordnungs-Objekte

Siehe Alle Oberflächen-Objekte

GroupBox



Notebook



NotebookPage



GroupSplit



GroupTile



Scrollbar



Divider



Kontakt

GroupBox



Gruppe

Liste,

Siehe Eigenschaften,
Ereignisse

Die Gruppe dient der Zusammenfassung von Objekten, um deren logische Zusammengehörigkeit auszudrücken. Über die Eigenschaft Caption kann eine Gruppenüberschrift angegeben werden.



Ob untergeordnete Objekte den Rahmen der Groupbox überdecken, kann über die Option WinAppExtTransparentBkg der Eigenschaft wpFlagsExt des App-Objektes definiert werden.

Kontakt

Notebook



Notizbuch

Liste,

Siehe Eigenschaften,
Ereignisse

Das Notebook eignet sich besonders gut zur Darstellung von Informationen, die in logisch zusammenhängenden Gruppen organisiert werden können. Zur Laufzeit kann der Wechsel der Notizbuchseiten durch Klicken mit der Maus auf den Registerreiter oder durch die Tastenkombination + bzw. + + erfolgen.

Im Designer kann durch Klick mit der rechten Maustaste auf den Registerreiter dessen Kontextmenü gestartet werden. In diesem Menü können Registerseiten eingefügt und geändert werden.

Das standardmäßige Verhalten des Notebook-Objekts kann über die Eigenschaft Flags geändert werden.

Bei der Anzeige von Bildern auf den Registerreitern (siehe ImageTile und ImageTileUser) wird die Angabe der Konstanten WinImgLeftTextRight bei der Eigenschaft ImageOption nicht unterstützt.

Kontakt

NotebookPage

 Seite eines Registerreiter

Liste,

Siehe Eigenschaften,

Ereignisse

Dieses Objekt hat als Vaterobjekt immer ein Notebook.

Der Titel der Seite wird mit der Eigenschaft Caption gesetzt.

Auf den Registerreitern können Bilder (siehe ImageTile und ImageTileUser) angezeigt werden. Die Größe der mitgelieferten Bilder wird über die Eigenschaft TileSize für WinTypeNotebook definiert.

Sollen benutzerdefinierte Bilder angezeigt werden, müssen die Eigenschaften TileNameNormal und TileNameSelected des Notebook-Objektes gesetzt werden. Die Bilder können in der Ressourcenverwaltung als Kachelgrafik für Schaltflächen- und Listenobjekte importiert werden.

Kontakt

GroupSplit



Objekt zur dynamischen Aufteilung des Fensterbereichs

Liste,

Siehe Eigenschaften,

Ereignisse,

GroupTile

Das GroupSplit-Objekt erlaubt die dynamische Aufteilung des Darstellungsbereiches in bis zu vier Einzelbereiche. Jeder Einzelbereich besteht aus einem GroupTile-Objekt, das im GroupSplit-Objekt angeordnet wird. Zur Laufzeit können die untergeordneten -Objekte vom Benutzer vergrössert oder verschoben werden.

Beim Setzen von Eigenschaften des GroupSplit-Objekts ist je nach Eigenschaft folgendes zu beachten:

StyleGroup

Es darf kein maximiertes GroupTile-Objekt geben.

Es darf kein GroupTile-Objekt geschlossen sein.

AreaMarginLeft, AreaMarginTop, AreaMarginRight, AreaBottom

Der übergebene Wert muss größer oder gleich Null betragen.

Vertical

Es darf kein maximiertes GroupTile-Objekt geben.

Es darf kein GroupTile-Objekt geschlossen sein.

GroupTile



Dynamischer Einzelbereich

Liste,

Siehe Eigenschaften,

Ereignisse,

GroupSplit

Das GroupTile-Objekt stellt einen Einzelbereich des GroupSplit-Objekts dar. Es ist wiederum in der Lage weitere Objekte aufzunehmen. Auch das Einfügen eines weiteren GroupSplit-Objekts in das GroupTile-Objekt ist möglich.

Die einem GroupTile-Objekt untergeordneten Objekte können über die Eigenschaft Grouping so angeordnet werden, dass sie den gesamten zur Verfügung stehenden Raum des Objekts einnehmen. Die untergeordneten Objekte werden bei Größenänderungen des GroupTile-Objekts automatisch angepasst.

GroupTile-Objekte besitzen (ähnlich zu Frame-Objekten) eine Titelleiste, über die sie verschoben, maximiert und geschlossen werden können. Wird ein GroupTile geschlossen, wird dessen Visible-Eigenschaft auf false gesetzt. Durch das Setzen dieser Eigenschaft auf true kann das Objekt wieder sichtbar gemacht werden.

Besitzt ein untergeordnetes Objekt eines GroupTile-Objekts den Tastaturfokus, wird die Titelleiste aktiviert dargestellt, und das GroupTile wird als aktiv bezeichnet.

Beim Setzen von Eigenschaften des GroupTile-Objekts ist je nach Eigenschaft folgendes zu beachten:

- Width und Height

Das GroupTile-Objekt darf nicht maximiert sein. Innerhalb des GroupSplit-Objekts müssen alle GroupTile-Objekte sichtbar sein. Der zugewiesene Wert darf nicht größer 10000 sein.

- AreaId

Das GroupTile-Objekt darf nicht maximiert sein. Es darf kein geschlossenes GroupTile-Objekt existieren. Der übergebene Wert muss mindestens Null sein und darf die Anzahl der im GroupSplit-Objekt angezeigten GroupTile-Objekte nicht übersteigen.

- FlagsTitlebar

Das GroupTile-Objekt darf nicht maximiert sein. Die Optionen können ausschließlich in Kombination mit der Option WinTitlebarCaption gesetzt werden.

- Visible

Die Eigenschaft AutoUpdate des GroupSplit-Objekts muss auf true gesetzt sein. Ein maximiertes GroupTile-Objekt kann nicht unsichtbar gesetzt werden. Die Eigenschaft AreaId muss einen gültigen Wert besitzen.

- AreaWidthMin, AreaWidthMax, AreaHeightMin und AreaHeightMax Der übergebene

Wert muss ≥ -1 betragen.

Kontakt

Scrollbar

 **Scrollbar**es Containerobjekt

[Liste](#), [Eigenschaften](#),

Siehe [Ereignisse](#),

[WinBoxScrollVisible\(\)](#)

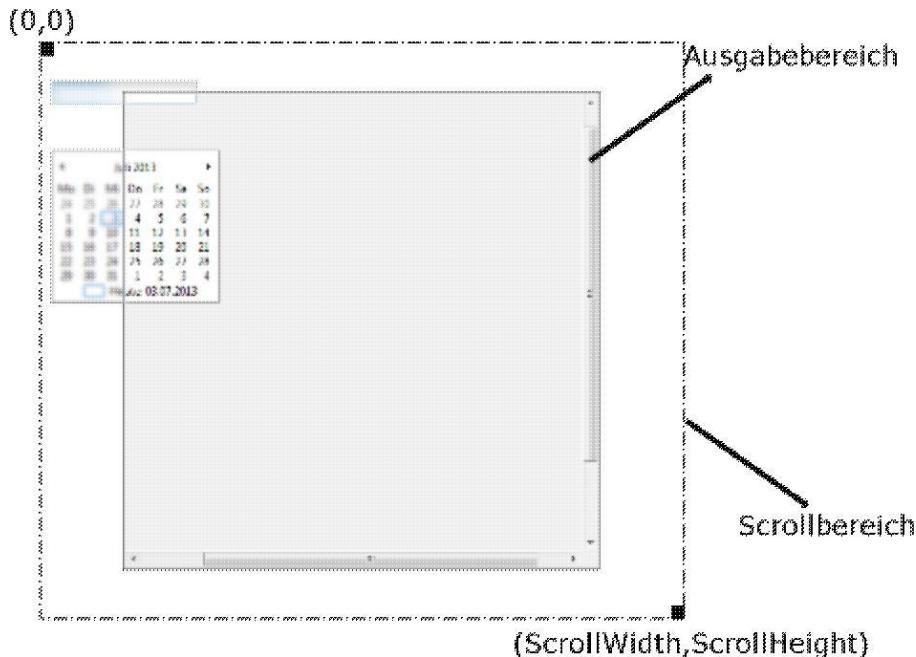
Das Scrollbox-Objekt ist ein Container-Objekt (ähnlich der Groupbox) zur Aufnahme von untergeordneten Objekten. Das Scrollbox-Objekt besitzt einen Scrollbereich, dessen Größe durch die Eigenschaften ScrollWidth und ScrollHeight definiert wird. Ist der Scrollbereich größer als der Ausgabebereich der Scrollbox (definiert durch die Objektgröße), dann werden Scrollbars angezeigt, mit denen der Scrollbereich verschoben werden kann. Über die Eigenschaften ScrollLeft und ScrollTop kann eine vorgegebene Position des Scrollbereichs eingestellt werden. Die der Scrollbox untergeordneten Objekte werden an den Grenzen des Scrollbereichs geclipt.

Die Eigenschaften ScrollLeft und ScrollTop können auch gesetzt werden, wenn das Scrolling durch die Eigenschaft SbarStyle unterbunden wurde.

Mit dem Befehl [WinBoxScrollVisible\(\)](#) kann ein Oberflächenobjekt in den sichtbaren Bereich gescrollt werden.

Die Scrollgeschwindigkeit bei Verwendung des Mausrads berücksichtigt die Windows-Einstellung in der Systemsteuerung (Mauseinstellungen).

Beispiel für ein Scrollbox-Objekt:



Divider



Trennlinie

Liste,

Siehe Eigenschaften,

Ereignisse

Das Divider-Objekt kann dazu verwendet werden, Ausgabebereiche zu unterteilen und somit übersichtlicher zu gestalten.

COM-Objekte

Objekte in der COM-Palette

Siehe Alle Oberflächen-Objekte

CtxOffice



CtxAdobeReader



CtxDocEdit



Kontakt

CtxOffice

 **Kontroll-Objekt für Microsoft Office-Dokumente**

[Liste](#),

[Eigenschaften](#),

Siehe [Ereignisse](#),

[InstallCtxOffice](#),

[COM-Befehle](#)

Über dieses Objekt kann in einem Dialog ein Microsoft Office-Dokument oder eine Karte von Microsoft MapPoint angezeigt und über die [COM-Schnittstelle](#) gesteuert werden. Damit ein Dokument angezeigt werden kann, muss die Office-Erweiterung über die Eigenschaft [InstallCtxOffice](#) installiert sein.

Ist bei der Initialisierung des Objekts ein Fehler aufgetreten, kann dieser mit der Anweisung [ComInfo\(\)](#) ermittelt werden.

In dem Objekt kann in der Eigenschaft [FileName](#) ein bestehendes Dokument angegeben oder ein neues Dokument angelegt werden.

Der Deskriptor des Objekts kann dazu verwendet werden, um Methoden des COM-Objekts aufzurufen (siehe [ComCall\(\)](#)) oder Eigenschaften der Oberfläche und des COM-Objekts zu setzen bzw. abzufragen.

Beispiel:

```
$CtxOffice->wpFileName # '*' + _Sys->spPathMyDocuments + '\MyDocument.doc';// Set document to rea
```

Kontakt

CtxAdobeReader



Kontroll-Objekt des Adobe Readers

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#),

[COM-Befehle](#)

Über diese Objekt kann der Adobe Reader gesteuert werden. Der Deskriptor des Objekts kann dazu verwendet werden, um Methoden des COM-Objekts aufzurufen (siehe [ComCall\(\)](#)) oder Eigenschaften der Oberfläche und des COM-Objekts zu setzen bzw. abzufragen.

Ist bei der Initialisierung des Objekts ein Fehler aufgetreten, kann dieser mit der Anweisung [ComInfo\(\)](#) ermittelt werden.



Damit dieses Objekt die PDF-Datei anzeigen kann, muss Adobe Reader installiert sein.

Beispiel:

```
$CtxAdobeReader->ComCall('setLayoutMode', 'SinglePage');$CtxAdobeReader->wpFileName # '*' + _Sys-
```

Kontakt

CtxDocEdit



Objekt zu Textverarbeitung

[Liste](#),

[Eigenschaften](#),

Siehe [Ereignisse](#),

[InstallCtxDocEdit](#),

[CtxDocEdit-Befehle](#),

[COM-Befehle](#), [Blog](#)

Über dieses Objekt können Texte verarbeitet werden. Das Objekt kann über die [COM-Schnittstelle](#) gesteuert werden. Damit ein Dokument angezeigt werden kann, müssen die [Gemeinsamen Komponenten](#) auf dem Rechner installiert und die DocEdit-Erweiterung über die Eigenschaft [InstallCtxDocEdit](#) eingerichtet sein.

Ist bei der Initialisierung des Objekts ein Fehler aufgetreten, kann dieser mit der Anweisung [ComInfo\(\)](#) ermittelt werden.

In dem Objekt kann in der Eigenschaft [FileName](#) ein bestehendes Dokument angegeben werden. Standardmäßig wird ein leeres Dokument beim Start angezeigt. Mit dem COM-Befehl `$ctxDocEdit->cplResetContents` kann der aktuelle Inhalt geleert werden. Jedoch bleibt die Eigenschaft [FileName](#) bei dem Befehl gesetzt.

Der Deskriptor des Objekts kann dazu verwendet werden, um Methoden des COM-Objekts aufzurufen (siehe [ComCall\(\)](#)) oder Eigenschaften der Oberfläche und des COM-Objekts zu setzen bzw. abzufragen.



Eine Dokumentation der möglichen Befehle, Eigenschaften und Ereignissen ist auf der [Hersteller-Seite](#) des Moduls zu finden.

Beispiel:

```
// Dokument laden
$CtxDocEdit->wpFileName # '*' + _Sys->spPathMyDocuments + '\MyDocument.doc';// D
```

Weitere Objekte

Siehe Alle Oberflächen-Objekte

Hyperlink



Calendar



Progress



GanttGraph



RecNavigator



Slider



Kontakt

HyperLink

HyperLink

Liste,

Siehe Eigenschaften,

Ereignisse

Ein Hyperlink ist ein Bezeichner, hinter dem eine Aktion liegt. Die Aktion wird ausgeführt, wenn der Benutzer auf das Objekt klickt. Auf diese Weise kann zum Beispiel der Web-Browser gestartet und eine bestimmte Seite aufgerufen werden.

Die auszuführende Aktion ist in der Eigenschaft LinkText anzugeben.

Kontakt

Calendar



Kalender

[Liste](#),

Siehe [Eigenschaften](#),

[Ereignisse](#)

Im Kalender-Objekt wird ein vollständiger Monat angezeigt. Durch Anklicken des Monats in der Titelzeile des Objekts kann der Monat und durch Anklicken der Jahresangabe das Jahr ausgewählt werden.

Unter Windows Vista ist das Calendar-Objekt betriebssystemseitig funktionell erweitert worden. Durch Anklicken des Titelbereichs wird die Anzeige innerhalb der Calendar-Seite verändert. Nach dem ersten Klicken werden die Monate, anschließend die Jahre, dann Jahresbereiche angezeigt. Durch Doppelklicken eines Monats / Jahres oder Jahresbereich wird die vorhergehende Anzeige wieder hergestellt.

Die Größe des Objektes kann bei verschiedenen Betriebssystemen unterschiedlich ausfallen. Damit das Objekt unter jeder Plattform angezeigt wird, ohne das Bereiche des Objekts abgeschnitten werden, kann die Eigenschaft AutoSize gesetzt werden. Die Größe des Objektes wird dann so angepasst, dass ein Monat angezeigt werden kann.

Das Objekt kann manuell so vergrößert werden, dass mehrere Monate dargestellt werden. Dies sieht beispielsweise wie folgt aus:

November 2013							Dezember 2013						
Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So
28	29	30	31	1	2	3						1	
4	5	6	7	8	9	10	2	3	4	5	6	7	8
11	12	13	14	15	16	17	9	10	11	12	13	14	15
18	19	20	21	22	23	24	16	17	18	19	20	21	22
25	26	27	28	29	30		23	24	25	26	27	28	29
							30	31	1	2	3	4	5

Heute: 2013-11-01

Über die Eigenschaft StyleDisplay wird festgelegt, ob im Kalender die Kalenderwochen angezeigt werden.

Das Datum kann mit der Eigenschaft CaptionDate gelesen und gesetzt werden.

Durch die Eigenschaft StyleTheme wird beeinflusst, ob das Objekt in Theme-Darstellung WinStyleThemeSystem oder in den angegebenen Farben WinStyleThemeNone gezeichnet wird. Die Eigenschaft wirkt sich bei dem Objekt nur aus, wenn die gleiche Eigenschaft beim Application-Objekt auf WinStyleThemeSystem gesetzt ist.

Ist die Eigenschaft TabStop auf true gesetzt, kann das Datum mit Hilfe der folgenden Tastaturkommandos relativ zum eingetragenen Datum verändert werden:

Tastenkombination Veränderung relativ zum aktuellen Datum

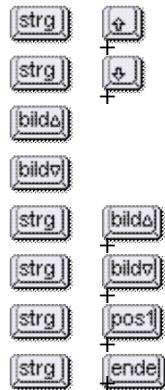


Gleicher Wochentag der vorherigen Woche



Gleicher Wochentag der nächsten Woche

Kontakt



- Vorheriger Tag
- Nächster Tag
- Gleicher Tag des Vorherigen Monats
- Gleicher Tag des nächsten Monats
- Gleiches Datum des vorhergehenden Jahres
- Gleiches Datum des nächsten Jahres
- 1. Tag des Monats des aktuellen Datums
- Letzter Tag des Monats des aktuellen Datums

Ist das Datum der 29.2. eines Schaltjahres und wird **[strg]** und **[bildΔ]** oder **[bildΔ]** gedrückt, wird auf den 28.2. des vorhergehenden bzw. nachfolgenden Jahres positioniert.



Die Tasten **[↑]**, **[↓]**, **[Δ]** und **[Δ]** können nur verwendet werden, wenn die Eigenschaft **EmulateKeys** des übergeordneten Frame-Objekts auf **false** gesetzt ist. **[↑]** und **[↓]** haben dann die gleiche Wirkung wie **[strg]+[↑]** bzw. **[strg]+[↓]**.

Progress

 Fortschrittsanzeige

[Liste](#),

[Eigenschaften](#),

Siehe [Ereignisse](#),

[Universelles](#)

[Modul \(Blog\)](#)

Mit diesem Objekt kann der Fortschritt einer Operation angezeigt werden.

Der Fortschritt des Balkens kann mit den Eigenschaften [ProgressPos](#) und [ProgressMax](#) gesetzt werden.

GanttGraph



Gantt-Diagramm

[Liste](#),
[Eigenschaften](#),
[Ereignisse](#),
[Befehle](#),
[PrtGanttGraph](#),

Siehe [Darstellung](#)

[von](#)
[Kennzahlen](#)
[\(Blog\)](#),
[Verwendung](#)
[\(Blog\)](#)

Mit diesem Objekt lassen sich zeitliche Abläufe darstellen. Das Gantt-Diagramm besteht aus mindestens zwei Achsen (eine horizontale und eine vertikale Achse) sowie einer Anzahl von Zellen und Intervallen.

Die Achsen werden mit dem [GanttGraph-Editor](#) erstellt und verändert.

Das Anlegen und Entfernen der Zeitintervalle erfolgt über die Befehle

[WinGanttIvlAdd\(\)](#) und [WinGanttIvlRemove\(\)](#).

Kontakt

Axis



x- und y-Achse

Siehe [Liste](#),

[Eigenschaften](#)

Ein [GanttGraph](#) besteht aus einer oder mehrerer x- und y-Achsen. Die Achsen werden mit dem [GanttGraph-Editor](#) erstellt und verändert.

Kontakt

Box



Box-Objekt

Siehe [Liste](#),

[Eigenschaften](#)

Bei einem Box-Objekt handelt es sich um einen farblich oder durch eine Schraffur hervorgehobenen Bereich innerhalb eines [Gantt-Diagramms](#).

Das Anlegen eines Box-Objekts geschieht durch den Befehl [WinGanttBoxAdd\(\)](#).

Kontakt

Interval



Intervall-Objekt

Siehe [Liste](#),

[Eigenschaften](#)

Ein [GanttGraph](#) besteht aus einem oder mehreren Intervall-Objekten. Das Anlegen und Entfernen der Intervalle erfolgt über die Befehle [WinGanttIvlAdd\(\)](#) und [WinGanttIvlRemove\(\)](#).

Kontakt

Line



Line-Objekt

Siehe [Liste](#),

[Eigenschaften](#)

Bei einem Line-Objekt handelt es sich um eine Hilfslinie innerhalb eines [Gantt-Diagramms](#).

Das Anlegen eines Line-Objekts geschieht durch den Befehl [WinGanttLineAdd\(\)](#).

View



View-Objekt

Siehe [Liste](#),

[Eigenschaften](#)

Ein [GanttGraph](#) besitzt mindestens ein View-Objekt. In dieser Ansicht werden die Intervalle des GanttGraphen dargestellt.

Ist die Eigenschaft [SplitStyle](#) nicht auf [WinSplitNone](#) gesetzt, kann über die "Splitter" (über bzw. links von den Scrollbars) die Ansicht geteilt werden. Auf diese Weise entstehen weitere View-Objekte (maximal vier). Der Deskriptor eines View-Objektes kann mit dem Befehl [WinInfo\(\)](#) ermittelt werden.

```
tHdlView # <Obj>->WinInfo(_WinObject, <View-Nummer>, _WinTypeGanttView);
```

In <Obj> wird der Deskriptor des GanttGraphen und in <View-Nummer> die Nummer der Ansicht übergeben. Ist die Ansicht nicht geteilt, existiert nur die Ansicht mit der Nummer 1. Alle weiteren Nummern werden von links nach rechts und von oben nach unten vergeben. Ist der [GanttGraph](#) horizontal in zwei Ansichten aufgeteilt, hat die obere Ansicht die Nummer 1, während die untere Ansicht die Nummer 2 besitzt.

Der ermittelte Deskriptor ist solange gültig, wie diese Views existieren. Der Benutzer kann durch Verschieben der "Splitter" die Ansicht so verändern, dass nur noch ein View vorhanden ist. Alle anderen Deskriptoren sind dann nicht mehr gültig.

Kontakt

RecNavigator

 Schaltflächenleiste für Datensatzoperationen

Liste,

Siehe Eigenschaften,

Ereignisse

Dieses Objekt beinhaltet alle Standard-Datensatzoperationen. Die Datensatzoperationen beziehen sich auf die Datei, die in der Eigenschaft DbFileNo angegeben wurde.

Die Schaltflächen haben folgende Bedeutung:



Vorigen oder nächsten Datensatz lesen



Liest fünf Datensätze nach vorne oder hinten



Ersten oder letzten Datensatz lesen



Neuen Datensatz anlegen



Bestehenden Datensatz löschen



Gesperrten Datensatz zurückschreiben



Datensatz sperren

Die Eigenschaft StyleConsole entscheidet über den Funktionsumfang des RecNavigators.

Wird über den RecNavigator ein neuer Datensatz gelesen, wird nach dem Lesen das Ereignis EvtDbFldUpdate beim Frame-Objekt (Frame oder MdiFrame) ausgelöst. Über das Ereignis kann in einem Frame in Abhängigkeit des gelesenen Datensatzes zusätzliche Informationen zum Beispiel aus anderen Datensätzen oder externen Dateien zur Verfügung gestellt werden.

Über das Ereignis EvtMouseItem kann ermittelt werden, welche Schaltfläche des Objekts angeklickt wurde. Bei Leseoperationen wird das Ereignis nach der Datensatzoperation und vor der Übertragung der Felder in die Objekte ausgelöst. Der Rückgabewert des Ereignisses wird nicht ausgewertet. Bei Schreiboperationen wird das Ereignis vor der Datensatzoperation ausgelöst. Die Schreiboperation kann mit dem Rückgabewert false verhindert werden.

Das Ergebnis der letzten Datensatzoperation kann über die Eigenschaft Erg ermittelt werden.

Slider



Schieberegler
Liste,

Siehe Eigenschaften,
Ereignisse

Beim Slider-Objekt handelt es sich um ein Oberflächenobjekt zur Auswahl eines ganzzahligen Wertes aus einem vorgegebenen Bereich.

Das Slider-Objekt setzt sich zusammen aus dem Kanal, einem Schieberegler, Markierungen und einem Textfeld. Über die Eigenschaft Vertical kann die Ausrichtung des Slider definiert werden.

Der Kanal wird durch einen farbigen Balken dargestellt, über den der Schieberegler verschoben werden kann. Die einzelnen Bestandteile sind in folgendem Bild ersichtlich:



Die Werte des Kanals werden durch die Eigenschaften MinInt und MaxInt definiert. Die Eigenschaft CurrentInt definiert die aktuelle Position des Schiebereglers im Kanal. Steht der Schieberegler ganz links (bzw. oben) enthält CurrentInt den Wert von MinInt. Wird der Schieberegler nun nach rechts (bzw. unten) verschoben, erhöhen sich die Werte für CurrentInt, solange bis das rechte (bzw. untere) Ende des Kanals erreicht ist. Dort enthält die Eigenschaft CurrentInt den Wert von MaxInt. CurrentInt enthält also immer einen Wert im Bereich von MinInt bis MaxInt in Abhängigkeit von der Position des Schiebereglers im Kanal.

Standardmäßig erhöht sich CurrentInt immer um eins, wenn der Schieberegler um eine Position nach rechts bzw. unten verschoben wird. Dies kann durch die Eigenschaft TickCount geändert werden. Wird dieser ein Wert von zwei zugewiesen, erhöht sich CurrentInt entsprechend um zwei, wenn der Schieberegler eine Position nach rechts bzw. unten verschoben wird.

Bei der Bedienung des Sliders mit der Tastatur haben die Eigenschaften TickLineSize und TickPageSize eine besondere Rolle. Sie geben die Anzahl der Positionen an, um

die der Schieberegler bewegt wird, wenn die Pfeiltasten bzw. die / -Tasten betätigt werden. Dabei kann der Schieberegler nur Werte annehmen, die im TickCount liegen. Sind die Eigenschaften TickLineSize bzw. TickPageSize kleiner als TickCount, wird CurrentInt dennoch um den Wert aus TickCount verändert.

Die Ausrichtung der Markierungen kann über die Eigenschaft TickAlignment gesteuert werden. Standardmäßig werden keine Markierungen angezeigt.

Jedem Wert aus dem Wertebereich [MinInt, MaxInt] kann ein Text zugeordnet werden, der als HelpTip und/oder im Textfeld des Sliders angezeigt wird. Die Texte werden hierbei durch das Zeichen ';' in der Eigenschaft HelpTip eingetragen. Sind weniger Texte vorhanden als Werte im Wertebereich, wiederholen sich die Texte

Kontakt

entsprechend. Das Textfeld des Sliders wird über die Eigenschaft Caption definiert. Hierbei ist die Verwendung von Escape-Sequenzen erlaubt. '\$CUR' stellt den Text für die aktuelle Position im Textfeld dar. Daneben gibt es noch '\$MIN' und '\$MAX' für den Text zu MinInt und MaxInt. Die Ausrichtung des Textes im Textfeld geschieht über die Eigenschaften Justify und JustifyVert.

Die Eigenschaft ShowFocus definiert, ob ein Fokusrechteck dargestellt werden soll, wenn der Slider den Eingabefokus erhält. In der systemabhängigen Darstellung hat die Eigenschaft keine Auswirkung. Dort wird immer ein Rechteck dargestellt. Nur in der Modern Theme Style-Darstellung wird die Eigenschaft berücksichtigt.

Kontakt

Eigenschaften eines Slider-Objekts

Das Objekt Slider unterstützt folgende Eigenschaften Slider,

Siehe [Liste der Objekte](#).

Liste der

- [AlignGrouping](#)
- [AlignHeight](#)
- [AlignMarginBottom](#)
- [AlignMarginLeft](#)
- [AlignMarginRight](#)
- [AlignMarginTop](#)
- [AlignWidth](#)
- [Area](#)
- [AreaBottom](#)
- [AreaHeight](#)
- [AreaLeft](#)
- [AreaRight](#)
- [AreaTop](#)
- [AreaWidth](#)
- [Caption](#)
- [CurrentInt](#)
- [Custom](#)
- [Disabled](#)
- [Font](#)
- [FontParent](#)
- [Help](#)
- [HelpTip](#)
- [HelpTipJustify](#)
- [HelpTipSysFont](#)
- [HelpTipTimeDelay](#)
- [HelpTipTimeShow](#)
- [Justify](#)
- [JustifyVert](#)
- [MaxInt](#)
- [MenuNameCntxt](#)
- [MinInt](#)
- [Name](#)
- [ShowFocus](#)
- [StyleBorder](#)
- [StyleTheme](#)
- [TabPos](#)
- [TabStop](#)
- [ThemeMenuCntxtTileSize](#)
- [ThemeSetId](#)
- [TickAlignment](#)
- [TickInterval](#)
- [TickLineSize](#)
- [TickPageSize](#)
- [Vertical](#)

Kontakt

- Visible

Kontakt

Ereignisse eines Slider-Objekts

Das Objekt Slider unterstützt folgende Ereignisse.

Slider,

Liste der

Siehe Objekte,

Liste der

Ereignisse

- EvtChanged
- EvtHelpTip
- EvtMenuCommand
- EvtMenuContext
- EvtMenuInitPopup
- EvtMouse

Kontakt

Canvas



Anzeige und Interaktion vektorbasierter Grafikobjekte

[Liste](#), [Eigenschaften](#),

Siehe [Ereignisse](#),

[CodeLibrary-Beispiel](#)

"Canvas"

Das Canvas-Objekt dient zur Anzeige und Interaktion vektorbasierter Grafikobjekte.

Das Canvas-Objekt ist ein Container, der Grafikobjekte verwaltet. Ein Grafikobjekt kann mittels [WinCreate\(_WinTypeCanvasGraphic, ...\)](#) erstellt und gleichzeitig oder nachträglich mit [WinAdd\(\)](#) zum Canvas-Objekt hinzugefügt und per [WinRemove\(\)](#) bzw. [WinDestroy\(\)](#) wieder entfernt werden. Der von [WinCreate\(\)](#) zurückgegebene Deskriptor kann verwendet werden, um die Eigenschaften des Grafikobjektes zu setzen oder zu lesen. Die Form der erzeugten Grafikobjekte kann über die Eigenschaft [FormType](#) des jeweiligen Objektes gesetzt werden.

Die Ausgabe der Grafikobjekte findet innerhalb der Ausgabefläche des Canvas-Objektes statt. Liegen Grafikobjekte außerhalb des sichtbaren Bereiches, zeigt das Canvas-Objekt Scrollbalken an, um die Grafikobjekte in den sichtbaren Bereich verschieben zu können. Die Zeichenfläche wird in geräteunabhängigen, logischen Einheiten bereitgestellt. Für die Umrechnung von Einheiten (z. B. Millimeter, Zentimeter) in logische Einheiten kann der Befehl [PrtUnitLog\(\)](#) verwendet werden.

Standardmäßig zeigt das Canvas-Objekt ein Raster an. Dieses kann über die Eigenschaften [GridWidth](#) und [GridHeight](#) ausgeblendet (einer der Werte = 0) oder in der Größe in logischen Einheiten definiert werden. Zudem werden am oberen und linken Rand standardmäßig Liniale angezeigt. Diese können mit der Eigenschaft [Ruler](#) ausgeblendet werden.

Beispiel:

```
// Canvas-Objekt erstellen und Objektgröße setzen  
cnvCanvas # WinCreate(_WinTypeCanvas, 'cnvCanvas',
```

Benutzerinteraktion

Das Canvas-Objekt unterstützt das Selektieren von [CanvasGraphic](#)-Objekten. Hierfür stehen mehrere Möglichkeiten zur Verfügung:

- Auswahl mit der Maus

Eine Auswahl kann aufgespannt werden, indem mit der linken Maustaste (☞) in einen freien Bereich des Canvas-Objektes geklickt und die Maustaste gehalten wird. Beim Bewegen der Maus wird ein rechteckiger Bereich aufgespannt. Wird die linke Maustaste losgelassen, werden alle [CanvasGraphic](#)-Objekte selektiert, die sich vollständig innerhalb des rechteckigen Bereiches befinden.

Ein nicht ausgewähltes [CanvasGraphic](#)-Objekt kann der Auswahl hinzugefügt werden, indem es mit [strg] + ☞ angeklickt wird. Auf diese Weise kann das Objekt auch wieder aus der Auswahl entfernt werden. Ein Mausklick ohne Zusatztaste auf ein nicht ausgewähltes [CanvasGraphic](#)-Objekt entfernt die

Kontakt

bereits vorhandene Auswahl und wählt das geklickte Objekt aus.

Ausgewählte CanvasGraphic-Objekte werden mit einem Selektionsrahmen mit acht Ankerpunkten dargestellt. Wird die Maus über einen Ankerpunkt geführt, verändert sich der Mauszeiger entsprechend. Klickt der Benutzer mit  auf einen Ankerpunkt und hält diese gedrückt, führen Bewegungen mit der Maus zu einer Größenänderung aller ausgewählten CanvasGraphic-Objekte. Die ausgewählten Objekte können auch verschoben werden. Dies geschieht, indem der Mauszeiger in ein bereits ausgewähltes Objekt geführt wird. Jetzt verändert

sich der Mauszeiger und zeigt ein Verschieben-Symbol an. Wird nun die  gedrückt und gehalten, werden alle ausgewählten Objekte verschoben.

Alle Operationen (Auswahl aufspannen, Größe ändern, Verschieben) können bei gehaltener linker Maustaste abgebrochen werden, indem die -Taste betätigt wird.



Eine Interaktion im Canvas-Objekt mit der Maus ist nur möglich, wenn die Eigenschaft ReadOnly des Objektes auf false gesetzt ist (default). Die Auswahl von Objekten mit der Maus wirkt sich auf die Eigenschaft SelData aus (siehe Auswahl per Programm-Code).

- Auswahl per Programm-Code

Die Mehrfachauswahl kann prozedural über die Eigenschaft SelData verändert werden. Die Eigenschaft enthält alle gegenwärtig im Canvas-Objekt ausgewählten CanvasGraphic-Objekte. Werden Objekte vom Anwender über Maus selektiert oder deselektiert, wird automatisch die Eigenschaft SelData aktualisiert. Zum prozedurealen Hinzufügen und Entfernen von CanvasGraphic-Objekten wird der Deskriptor des CanvasGraphic-Objektes bei WinMsdInsert() bzw. WinMsdDelete() angegeben. Zur Aktualisierung der Mehrfachauswahl muss anschließend WinUpdate(WinUpdState) durchgeführt werden.

Kontakt

CanvasGraphic

 **Grafikobjekt zur Darstellung in einem Canvas-Objekt Liste, Eigenschaften,**
Siehe CodeLibrary-Beispiel
"Canvas"

Grafikobjekte stellen den Inhalt eines Canvas-Objektes dar. Sie können mit dem Befehl WinCreate(WinTypeCanvasGraphic, ...) angelegt werden.

Über die Eigenschaft FormType wird die Form des Grafikobjektes definiert. Die Position und Größe wird mit der Eigenschaft Area in logischen Einheiten definiert. Für die Umrechnung von Einheiten (z. B. Millimeter, Zentimeter) in logische Einheiten kann der Befehl PrtUnitLog() verwendet werden.

Farbgebung

Die Eigenschaft ColBkg definiert die Füllfarbe des Grafikobjektes. Eine transparente Darstellung kann über WinColorOpacitySet() erreicht werden. Die Grafikobjekte können mit einem Rahmen versehen werden. Die Stärke des Rahmens in logischen Einheiten, wird über die Eigenschaft BorderWidth definiert. Die Füllfarbe des Rahmens wird über die Eigenschaft BorderColFg definiert. Auch hier ist die Angabe einer teiltransparenten Farbe durch WinColorOpacitySet() möglich.



Die Füll- (ColBkg) und Rahmenfarbe (BorderColFg) werden beim FormType WinFormTypeLine ignoriert. Text- und Linien-Farbe werden über die Eigenschaft ColFg definiert. Hierbei werden teiltransparente Farben nicht berücksichtigt.

Inhalt

Als Inhalt können sowohl Texte als auch Grafiken dargestellt werden.

Der anzuzeigende Text wird über die Eigenschaft Caption definiert. Dieser kann mehrzeilig sein. Wortumbrüche können über die Eigenschaft WordBreak aktiviert werden. Die Eigenschaft Vertical gibt an, ob der Text vertikal oder horizontal (default) angezeigt wird. Die Textausrichtung geschieht über die Eigenschaften Justify und JustifyVert.

Ein Bild kann über die Eigenschaft PictureName gesetzt werden. Ein Stern ('*') als Präfix kennzeichnet ein extern vorliegendes Bild. Die Anzeige des Bildes erfolgt über die Eigenschaft PictureMode. Die Eigenschaft Vertical definiert, ob das Bild gekippt dargestellt wird oder nicht.

Besonderheiten je nach FormType

WinFormTypeRectangle / WinFormTypeSquare:

Bei der Ausgabe des rechteckigen Bereiches (WinFormTypeRectangle, WinFormTypeSquare) können abgerundete Ecken über die Eigenschaft Justify und Radius gesetzt werden. Die Eigenschaft definiert den Radius der abgerundeten Ecken in logischen Einheiten.

Kontakt

WinFormTypeLine:

Bei der Linie werden Füll- (ColBkg) und Rahmenfarbe (BorderColFg) ignoriert. Die Farbe der Linie wird über die Eigenschaft ColFg gesetzt. Die Linienstärke in logischen Einheiten definiert die Eigenschaft LineWidth. Die Linie kann mit einem Pfeiltyp für den Start- bzw. Endpunkt versehen werden. Hierfür existiert die Eigenschaften LineCapStart und LineCapEnd.

Die Ausrichtung der Linie kann horizontal (Vertical = false) oder vertikal (Vertical = true) sein. Die Ausrichtung der Linie (links, zentriert, mittig) geschieht über die Eigenschaft LineJustify. Bild und Text werden in Abhängigkeit von der Ausrichtung links / über bzw. rechts / unter der Linie angezeigt.

Kontakt

Eigenschaften eines CanvasGraphic-Objekts

Das CanvasGraphic-Objekt unterstützt folgende Eigenschaften.

CanvasGraphic,

Liste der

Siehe Objekte, Liste

der

Eigenschaften

- Area
- AreaBottom
- AreaHeight
- AreaLeft
- AreaRight
- AreaTop
- AreaWidth
- BorderColFg
- BorderWidth
- Caption
- ColBkg
- ColFg
- Custom
- Font
- FontParent
- FormType
- HelpTip
- Justify
- JustifyVert
- LineCapEnd
- LineCapStart
- LineJustify
- LineWidth
- ModeEffect
- Name
- PictureMode
- PictureName
- Radius
- Rotation
- Vertical
- WordBreak

Kontakt

Eigenschaften eines Canvas-Objekts

Das Objekt Canvas unterstützt folgende Eigenschaften.

Canvas, Liste
der Objekte,

Siehe

Liste der

- AlignGrouping
- AlignHeight
- AlignMarginBottom
- AlignMarginLeft
- AlignMarginRight
- AlignMarginTop
- AlignWidth
- AutoUpdate
- ColBkg
- ColBkgApp
- ColBoundary
- ColFg
- Custom
- Font
- FontParent
- FormHeight
- FormWidth
- GridHeight
- GridWidth
- Name
- ReadOnly
- Ruler
- SelData
- Unicode
- Unit
- ZoomFactor

Kontakt

Ereignisse eines Canvas-Objekts

Das Objekt Canvas unterstützt folgende Ereignisse.

Canvas,

Liste der

Siehe Objekte,

Liste der

Ereignisse

- EvtFocusInit
- EvtFocusTerm
- EvtHelpTip
- EvtMenuCommand
- EvtMenuContext
- EvtMenuItemPopUp
- EvtMenuItem

Kontakt

Druckformular-Objekte

Ein Druckformular besteht aus ein oder mehreren Objekten.

Siehe Alle

Druckformular-Objekte

Die Objekte untergliedern sich in folgende Bereiche:

- Druck-Objekte
- Drucker-Objekte
- Seiten-Objekte
- Tabellen-Objekte
- Formular-Objekte

Die maximale Größe der Objekte beträgt 44x44 inch (ca. 112 cm).

Formular-Objekte

Siehe Alle Druckformular-Objekte

PrintDoc



PrintDocRecord



PrintForm



PrintFormList



PrintForm



PrintForm

[Liste](#),

[Eigenschaften](#),

Siehe [Beispiel](#),

[PrintFormList](#),

[Blog](#)

Das PrintForm-Objekt dient dem dynamischen Aufbau von Druckdokumenten ([PrintDoc](#)-Objekt). Hierzu gibt es spezielle Prozedurbefehle, die die im PrintForm-Objekt enthaltenen Druckobjekte einem PrintDoc-Objekt hinzufügen.

Das PrintForm-Objekt wird dann verwendet, wenn ein Druckjob aus kleineren oder sehr unterschiedlichen Elementen aufgebaut werden soll. Zum Beispiel eine Rechnung oder eine Liste, die auf Daten aus mehreren Tabellen zugreift.

Wird ein PrintForm-Objekt im Designer erstellt, ist in der Objektpalette nur der Registerreiter "Druck" vorhanden, und nur die dort enthaltenen Objekte können dem Objekt hinzugefügt werden.