

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->spPathTen
```

Mögliche Laufzeitfehler:

_ErrValueInvalid Im Parameter (int1) wurde ein ungültiger Typ oder in (int5) ein ungültiger Wert angegeben.

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**

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:

<u>ErrValueInvalid</u>	In (alpha1) wurde eine leere Zeichenkette oder in (int2) ein ungültiger Wert angegeben. Der Laufzeitfehler wird auch generiert, wenn <u>ChartFormatAuto</u> angegeben ist, der Dateiname jedoch keine gültige Endung besitzt.
<u>ErrHdlInvalid</u>	Bei dem in (obj) übergebenen Deskriptor handelt es sich nicht um einen gültigen <u>Chart</u> -Deskriptor oder bei (handle3) nicht um ein <u>Memory</u> -Objekt.
<u>ErrValueRange</u>	Das übergebene <u>Memory</u> -Objekt hat nicht genug Platz zum Schreiben aller Daten.

obj -> ChartClose()



Chart-Grafik schließen

obj Deskriptor des
 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.

obj	Deskriptor eines <u>Chart</u> - oder <u>ChartData</u> -Objekts
int1	Maximale Anzahl an Werten Informationstypen (optional)
	_ChartDataValue Datenmenge
	_ChartDataLabel Beschriftungen
int2	_ChartDataColor Füllfarben für Datenbereiche
	_ChartDataExtra Zusätzliche alphanumerische Daten

Siehe [Verwandte Befehle](#), [ChartDataClose\(\)](#), [ChartDataAdd\(\)](#)

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.

Neben der Datenmenge gibt es noch weitere Datentypen. Folgende Konstanten können als Datentypen angegeben werden:

Mit dieser Konstante wird das Objekt zum Transport der Datenmenge vorbereitet. Da jedes Chart-Objekt eine Datenmenge zur Anzeige benötigt, muss diese Konstante beim Erzeugen des ChartData-Objekts nicht angegeben werden. Beim Hinzufügen von Daten ist das der Default-Wert.

Mit dieser Konstante wird das Objekt zum Transport von Beschriftungen für die einzelnen Daten vorbereitet. Diese Beschriftungen werden neben den Sektoren (Darstellung als Torten-Diagramm, `_ChartPie`) oder unter der X-Achse (Darstellung als Koordinaten-Diagramm, `_ChartXY`) angezeigt.

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`)

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:

```
tHdlChart # ChartOpen(_ChartPie, 400, 300);tHdlChartData # tHdlChart->ChartDataOpen(20);...
```

Kontakt

Es werden Daten, Bezeichner und Farben übermittelt:

```
tHdlChart # ChartOpen(_ChartPie, 400, 300);tHdlChartData # tHdlChart->ChartDataOpen(20, _ChartDat
```

Mehrere Datenreihen:

```
tHdlChart # ChartOpen(_ChartXY, 400, 300);// first rowtHdlChartData # tHdlChart->ChartDataOpen(20,
```

Datenreihen für Oberflächen-Diagramme:

```
tHdlChart # ChartOpen(_ChartSurface, 400, 300);tHdlChartDataMain # tHdlChart->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:

- | | |
|------------------------|---|
| <u>ErrHdlInvalid</u> | Bei dem übergebenen Deskriptor in (obj) handelt es sich nicht um einen gültigen Deskriptor eines <u>Chart</u> -Objekts. |
| <u>ErrValueInvalid</u> | Das Argument (int2) enthält einen ungültigen Wert oder das Argument (int1) ist negativ. |



obj -> ChartDataAdd(var1[, int2]) : int
Daten in ein ChartData-Objekt einfügen

obj Deskriptor auf ein
 ChartData-Objekt
var1 Einzufügende Daten
 Datentyp
 ChartDataValue Datenmenge
int2 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() entsprechen vorbereitet wurde.


Konnte die Anweisung ausgeführt werden, gibt sie den Wert ErrOk zurück. Steht kein ausreichender Speicher zur Verfügung, wird ErrOutOfMemory zurückgegeben.

Beispiel:

```
tChartData->ChartDataAdd('Frankfurt am Main', _ChartDataLabel);tChartData->ChartDataAdd(664838);t
```

Mögliche Laufzeitfehler:

<u>_ErrValueInvalid</u>	In (int2) wurde ein ungültiger Wert übergeben oder der angegebene Datentyp wurde nicht bei <u>ChartDataOpen()</u> spezifiziert. Der Laufzeitfehler wird auch generiert, wenn die Obergrenze der insgesamt möglichen Daten (siehe <u>ChartDataOpen()</u>) überschritten wurde.
<u>_ErrFldType</u>	In (var1) wurde ein nicht unterstützter Datentyp übergeben.

obj -> ChartDataSort([int1[, range2]]) 

: int

Daten im ChartData-Objekt sortieren

obj Deskriptor des ChartData-Objekts

Sortier-Optionen (optional)

 _ChartDataSortLabel Sortierung nach der
 Beschriftung

int1 _ChartDataSortLabelCI Sortierung nach der
 Beschriftung (ohne
 Unterscheidung der
 Groß-/Kleinschreibung)

 _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(_ChartDataSortLabel)
```

Als Rückgabewert wird immer ErrOk zurückgegeben.

Mögliche Laufzeitfehler:

Kontakt

- __ErrValueInvalid Der in (int1) angegebene Datentyp ist nicht im ChartData-Objekt enthalten oder der angegebene Bereich liegt nicht innerhalb der bei ChartDataOpen() angegebenen Anzahl der Elemente.
- __ErrHdlInvalid In (obj) wurde kein gültiger ChartData-Deskriptor angegeben.



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:

__ErrHdlInvalid 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

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

Resultat alpha Inhalt der
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.



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 befinden 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 Wurzelements 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 document
// attributes to element tParent->CteInsertNode('state', _XmlNodeAttribute, 'Hessen', _CteAttribute);}
```

Das Beispiel schreibt folgende Datei:

```
<?xml version="1.0" encoding="UTF-8"?><?xml-stylesheet type="text/xsl" href="style.xsl"?><!--Liste der Bundesländer--><state name="Hessen" value="Hessen" /></state></xml>
```

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
- XmlGetValueInt

- XmlLoad
- XmlOpenReader
- XmlOpenWriter
- XmlRead
- XmlSave
- XmlWrite

Konstanten

- XmlEndDocument
- XmlEndElement
- XmlErrorCode
- XmlErrorColumn
- XmlErrorLine
- XmlErrorText
- XmlGetAttribCount
- XmlGetDocType
- XmlGetEncoding
- XmlGetName
- XmlGetNodeDepth
- XmlGetType
- XmlGetValue
- XmlGetVersion
- XmlIsStandalone
- XmlIsValid
- XmlLoadHugeTextNode
- XmlNodeAttribute
- XmlNodeCDATA
- XmlNodeComment
- XmlNodeDocument
- XmlNodeDocumentType
- XmlNodeElement
- XmlNodeProcessingInstruction
- XmlNodeText
- XmlOpenReaderDefault
- XmlOpenReaderDTDValidate
- XmlOpenWriterDefault
- XmlOpenWriterOverwrite
- XmlOpenWriterPure
- XmlReadAttribNode
- XmlReaderTypeAttribute
- XmlReaderTypeCDATA
- XmlReaderTypeComment
- XmlReaderTypeDocument
- XmlReaderTypeDocumentFragment
- XmlReaderTypeDocumentType
- XmlReaderTypeElement
- XmlReaderTypeEndElement
- XmlReaderTypeEntity
- XmlReaderTypeEntityReference
- XmlReaderTypeNotation

- XmlReaderTypePI
- XmlReaderTypeSigWhitespace
- XmlReaderTypeText
- XmlReaderTypeWhitespace
- XmlSaveDefault
- XmlSavePure
- XmlStartDocument
- XmlStartElement
- XmlWriteAttribute
- XmlWriteCDATA
- XmlWriteComment
- XmlWriteDocType
- XmlWriteElement
- XmlWritePI
- XmlWriteText



obj -> XmlClose()

Schließt ein XmlReader- bzw. 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:

_ErrHdlInvalid Deskriptor (obj) ist kein gültiger XmlReader- oder
XmlWriter-Deskriptor.

XmlError(int1) : alpha



Weitere Informationen zu eine 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.



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\(\)](#), [XmlGetValueInt\(\)](#)

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 (2)	Kodierung - Dieser Wert steht nach dem ersten XmlRead() zur Verfügung.
_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.



obj -> XmlGetValueInt(int1) : int

Ermittelt eine numerische Eigenschaft eines XML-Knotens oder einer -Datei

obj Deskriptor des XmlReader-Objektes

Zu ermittelnder Wert

int1

_XmlIsStandalone	Ist XML-Dokument Standalone
_XmlGetType	Typ des Knotens
_XmlGetNodeDepth	Ebenentiefe des Knotens
_XmlIsValid	Ist XML-Dokument valide
_XmlGetAttribCount	Anzahl der Attribute des Knotens

Resultat int Wert abhängig von der angegebenen Option oder ErrGeneric bei 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 <u>XmlRead()</u> zur Verfügung.
_XmlGetType (2)	Typ des Knotens _XmlReaderTypeElement (1) Element _XmlReaderTypeAttribute (2) Attribut _XmlReaderTypeText (3) Text _XmlReaderTypeCDATA (4) Daten _XmlReaderTypeEntityReference (5) Verweis auf Textbaustein _XmlReaderTypeEntity (6) Textbaustein _XmlReaderTypePI (7) Verarbeitungsanweisung _XmlReaderTypeComment (8) Kommentar _XmlReaderTypeDocument (9) Dokument _XmlReaderTypeDocumentType (10) Dokumenttyp _XmlReaderTypeDocumentFragment (11) Dokumentfragment _XmlReaderTypeNotation (12) Notation _XmlReaderTypeWhitespace (13) Leerzeichen _XmlReaderTypeSigWhitespace (14) Leerzeichen _XmlReaderTypeEndElement (15) Elementende
_XmlGetNodeDepth (3)	Ebenentiefe des Knotens

_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.

obj ->

XmlLoad(alpha1[,
int2[, handle3[,
alpha4[, handle5]]]]) :



int

XML-Daten lesen

obj Deskriptor eines
 CteNode-Objekts

alpha1 Pfad- und Dateiname
 Optionen (optional)

XmlLoadHugeTextNode Knoten
 mit
int2 Inhalten
 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 XmlError() 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.

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

Resultat handle Deskriptor des XmlReaders oder 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.

XmlOpenWriter(alpha1[, int2]) : handle
 XML-Datei zum sequenziellen Schreiben öffnen
 alpha1 Pfad und Name der XML-Datei



Optionen (optional)

	<u>_XmlOpenWriterDefault</u>	Datei formatiert schreiben
int2	<u>_XmlOpenWriterPure</u>	Datei unformatiert schreiben
	<u>_XmlOpenWriterOverwrite</u>	Existierende Datei überschreiben

Resultat handle Deskriptor des XmlWriters oder Fehlerwert 

Siehe Verwandte Befehle, XmlWrite(), XmlClose(),
XmlOpenReader()

Diese Anweisung öffnet die XML-Datei (alpha1) zum sequenziellen Schreiben.

Folgende Optionen (int2) können angegeben werden:

<u>_XmlOpenWriterDefault</u> (0)	Datei formatiert schreiben (Standard)
<u>_XmlOpenWriterPure</u> (1)	Datei unformatiert schreiben
<u>_XmlOpenWriterOverwrite</u> (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:

<u>_ErrGeneric</u>	Allgemeiner Fehler
<u>_ErrOutOfMemory</u>	Speicher konnte nicht angefordert werden
<u>_ErrFsi...</u>	Fehler beim Dateizugriff

Beispiel:

```
// XmlWriter öffnetXmlWriter # XmlOpenWriter('C:\File.xml', _XmlOpenWriterPure | _XmlOpenWriterC
```

Mögliche Laufzeitfehler:

_ErrValueInvalid Die angegebene Option (int2) ist ungültig.



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

Siehe Verwandte Befehle, XmlOpenReader(),
XmlGetValueInt(), 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 öffnet XmlReader # 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.

```
obj ->
XmlSave(alpha1[,
int2[, handle3[, int4[,
int5]]]) : int
XML-Daten schreiben
obj      Deskriptor des zu
        schreibenden Objekts
alpha1   Pfad- und Dateiname
        Optionen (optional)
        _XmlSaveDefault Formatierte
int2      Ausgabe
        _XmlSavePure   Unformatierte
        Ausgabe
handle3   Deskriptor eines
        Memory-Objekts (optional)
int4      Zielzeichensatz (optional)
int5      Quellzeichensatz (optional)
Resultat int Fehlerwert
Siehe    Verwandte Befehle,
        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.

Könnte die externe Datei nicht geschrieben werden, wird ein ErrFsi...-Fehlerwert zurückgegeben.

Beispiel:

```
sub WriteXML( aHdlMem          : handle;) local { tXMLDoc          : handle; tXMLItemRoot : handle;  
  // Kommentar tXMLItemRoot->CteInsertNode(' ', _XmlNodeComment, 'comment'); // Text tXMLItemRoot->CteInsertNode(' ', _XmlNodeText, 'comment');
```

Das Resultat sieht wie folgt aus:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><!DOCTYPE myRootElement SYSTEM "myDTD.dtd">  
<myRootElement>  
  <comment>  
  </comment>  
</myRootElement>
```

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.

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

_XmlStartElement Neues Element öffnen

_XmlEndElement Element schließen

int1 _XmlWriteElement Neues Element einfügen

_XmlWriteAttribute 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.

_XmlStartElement (3) Neues Element öffnen

Kontakt

	Argument1 Name
<code>_XmlEndElement</code> (4)	Element schließen Keine Argumente benötigt
<code>_XmlWriteElement</code> (5)	Neues Element einfügen Argument1 Name Argument2 Inhalt  Das Element kann keine Kindelemente oder Attribute enthalten.
<code>_XmlWriteAttribute</code> (6)	Attribut schreiben Argument1 Name Argument2 Wert
<code>_XmlWriteDocType</code> (7)	DTD-Verweis einfügen Argument1 Name Argument2 Speicherort einer Public-DTD Argument3 Speicherort einer System-DTD
<code>_XmlWritePI</code> (8)	Verarbeitungsanweisung einfügen Argument1 Ziel Argument2 Inhalt
<code>_XmlWriteText</code> (9)	Text einfügen Argument1 Text
<code>_XmlWriteCDATA</code> (10)	Daten einfügen Argument1 Daten
<code>_XmlWriteComment</code> (11)	Kommentar einfügen Argument1 Kommentar

 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 lesbares 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, bigint, 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.

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

obj ->

JsonLoad(alpha1[, ,
int2[, handle3]]) :

int

JSON-Daten lesen

obj Deskriptor eines
 CteNode-Objekts

alpha1 Pfad- und Dateiname

int2 0 - Reserviert (optional)

 Deskriptor eines
handle3 Memory-Objekts
 (optional)

Resultat int ErrOk oder
 Fehlerposition

Verwandte Befehle,

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 angegeben 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.

obj ->
 JsonSave(alpha1[,
 int2[, handle3[, int4[,
 int5]]]) : int
 JSON-Daten schreiben
 obj Deskriptor eines
 CteNode-Objekts
 alpha1 Pfad- und Dateiname
 Optionen (optional)
 _JsonSaveDefault Formatierte
 int2 Ausgabe
 _JsonSavePure Unformatierte
 Ausgabe
 handle3 Deskriptor eines
 Memory-Objekts (optional)
 int4 Zielzeichensatz (optional)
 int5 Quellzeichensatz (optional)
 Resultat int Fehlerwert 
 Siehe Verwandte Befehle, JSON,
 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 leer
```

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.

COM-Befehle

Befehle zur Kommunikation mit der COM-Schnittstelle

Beispiel, Liste

sortiert nach

Siehe Gruppen,
Alphabetische

Liste aller

Befehle

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-Variablentyp 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 zurückliefert.

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.

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.

obj -> ComArgGet(int1, ) : int

COM-Argument ermitteln

obj COM-Argument-Objekt

int1 Nummer des 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.

obj ->
 ComArgSet(int1, var2) 
 : int
 COM-Argument setzen
 obj COM-Argument-Objekt
 int1 Nummer des 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.



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 Variablennamen 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.



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.



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



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 startentComApp # ComOpen('Word.Application', _ComAppCreate);...// Microsoft Word
```

Die Konstante wdDoNotSaveChanged hat den Wert 0.

obj -> ComPropGet(alpha1, var2[, int3, ...,  int6]) : int


Eigenschaft eines COM-Objektes abfragen

obj COM-Objekt

alpha1 Name der Eigenschaft

var2 Wert der Eigenschaft

int3-int6 zusätzliche Informationen

Resultat int Fehlerwert 

Verwandte Befehle,

Siehe 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 abfragenArray->ComPropGet('Item', aValue, 1, 1);
```

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)


Siehe Verwandte Befehle, ComOpen(),
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-dimensionalen Arrays abfragenArray->ComPropGetText('Item', tHdlText, 1, 1);
```

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 O
```

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 O
```

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-dimensionalen Arrays setzentArray->ComPropSet('Item', 'Jahr', 1, 1);
```


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).

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.

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->cpiItem(1, 1) # 'Year'; tArray->cpiItem(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

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 Zellenkoordinate 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

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:

```
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.

Konstanten für COM-Befehle
Konstanten für die COM-Befehle
Siehe COM-Befehle

- ComAppCreate
- ComInfoErrCode
- ComInfoErrText

_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.

_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.

_ComInfoErrText

Fehlertext ermitteln

Wert 1

Siehe ComInfo()

Option bei ComInfo() - Mit dieser Option wird der von der über COM angesteuerten Applikation erzeugte Fehlertext ermittelt.

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.

obj ->

DdeCommand(int1, ) : int

DDE-Befehl absetzen

obj	Deskriptor des DDE-Kanals Funktionstyp <u>DdeReceive</u>	Themeninhalt abrufen
	<u>DdeSend</u>	Information übertragen
int1	<u>DdeExecute</u>	Kommando ausführen
	<u>DdeAdviseOn</u>	Aktualisierung einschalten
	<u>DdeAdviseOff</u>	Aktualisierung ausschalten
int2	Maximale Wartezeit	
alpha3	Elementname bzw. optionales Kommando	
Resultat	<u>int</u>	Fehlerwert 
Siehe	<u>Verwandte Befehle</u> , <u>DdeConnect()</u>	

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 Verbindungsaufbaus

Siehe Verwandte Befehle, DdeInit(),
 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

obj -> DdeDisconnect()



DDE-Verbindung abbauen

obj Deskriptor des
 DDE-Kanals

Verwandte

Siehe Befehle,

DdeConnect()

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

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

obj -> DdeInit(int1) : handle
 DDE-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)
```

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
int1 DdeServiceApp Applikation
 DdeServiceTheme Thema
Resultat alpha Serviceinformation
Siehe Verwandte Befehle,
 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

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

Siehe Verwandte Befehle,
 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

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 Themename (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 (alpha2) 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



obj -> DdeSetData(alpha1, int2)

Daten in den DDE-Sendepuffer übertragen

obj Deskriptor des
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) angehängen. Der Sendepuffer vergrößert sich automatisch nach Bedarf.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

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

Konstanten für DDE-Befehle

Siehe Alle
Befehle

<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

_DdeAdviseOff
Aktualisierungsnachricht
Wert 5

Verwandte

Siehe Befehle,

DdeCommand()

Option bei DdeCommand() - Aktualisierungsnachrichten ausschalten.

_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');
```

_DdeAppend

Daten anhängen

Wert 1

Verwandte

Siehe Befehle,

DdeSetData()

Option bei DdeSetData(). Die übertragenen Daten werden im Puffer an vorhandene Daten angehängt.

DdeCRLF
Übertragungsformat
Wert 2

Verwandte

Siehe Befehle,

DdeSetData()

Option bei DdeSetData() - Daten mit CR + LF anhängen

_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.

_DdeReceive
Themeninhalt
Wert 1

Verwandte

Siehe Befehle,

DdeCommand()

Option bei DdeCommand() - Themeninhalt abrufen

_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.

_DdeServiceApp

Applikation

Wert 1

Verwandte

Siehe Befehle,

DdeServiceData()

Option bei DdeServiceData() - Applikation ermitteln

_DdeServiceNext

Eintrag lesen

Wert 1

Siehe Verwandte Befehle,

DdeServiceRead()

Option bei DdeServiceRead() - nächsten Eintrag lesen

_DdeServicePos

Eintrag lesen

Wert 0

Siehe Verwandte Befehle,

DdeServiceRead()

Option bei DdeServiceRead() - Eintrag direkt lesen

Kontakt

_DdeServicePrev

Eintrag lesen

Wert 2

Siehe Verwandte Befehle,

DdeServiceRead()

Option bei DdeServiceRead() - vorherigen Eintrag lesen

_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

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

```
tDll # DllLoad('C:\c16\57\mydll');if (tDll <= 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. Desweiteren 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**

- **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_QueryPgifInfo()</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

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: instan
/* retrieve subrecord information by name */vERROR (C16API_DLL * C16_SbrInfoByName)( const vPHAN
/***** 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

```

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#de
#define C16ERR_BIN_NAME_INVALID              -851#define C16ERR_BIN_NO_PATH              -852#de
// 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 _FileAttrOemMark                    0x01#define _FileAttrTemp                    0x02#define
#define _RecFirst                          0x00000001#define _RecLast                      0x000
#define _RecCount                          0#define _RecCountPos                        5#define _RecC
#define _SelFirst                          0x01#define _SelLast                          0x02#define
#define _PgifType_Client                    0x0001 /* Client-type */#define _PgifTy
#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 umfangreicheren 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.

DllLoad(alpha1[,  alpha2]) : handle
DLL oder SO laden

alpha1 Pfad und Name der DLL
(Windows) oder des SO
(Linux)
alpha2 Name der
Einstiegsfunktion
(optional)

Resultat handle Deskriptor oder
_ErrGeneric

Siehe Verwandte Befehle,
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 geliefert 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.

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 ErrHdlInvalid.

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
- MailSmtplib
- MailSmtplibS
- MailSmtplibTls
- SckNoTlsV1
- SckTlsHigh
- SckTlsLow
- SckTlsMax
- SckTlsMed
- SckTlsSNI
- SmtplibBCC
- SmtplibCC
- SmtplibDiscard
- SmtplibFrom
- SmtplibPriority
- SmtplibReplyTo
- SmtplibSendNow
- SmtplibSubject
- SmtplibTo

- 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_QP

MailOpen(int1, alpha2[,
int3[, int4[, alpha5,
alpha6]]) : handle



Neue E-Mail erzeugen

Mail- / Authentifizierungstyp

<u>MailSmtp</u>	Unverschlüsselte Verbindung aufbauen
<u>MailSmtpTls</u>	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 speichern
<u>SckTlsMax</u>	Verschlüsselung mit maximalen Sicherheitsanforderungen
<u>SckTlsHigh</u>	Verschlüsselung mit hohen Sicherheitsanforderungen
<u>SckTlsMed</u>	Verschlüsselung mit durchschnittlichen Sicherheitsanforderungen
<u>SckTlsLow</u>	Verschlüsselung mit niedrigen Sicherheitsanforderungen
<u>SckNoTLSv1</u>	Verschlüsselung ohne Unterstützung von TLS 1.x
<u>SckTlsSNI</u>	Verwendung von Server Name Indication (SNI) beim verschlüsselten Verbindungsaufbau

int1

alpha2 Name oder IP-Adresse des
Mailservers / Name der EML-Datei

int3 SMTP-Port (optional)

int4 Timeout in Sekunden (optional)

alpha5 Benutzername (optional)

alpha6 Kennwort (optional)

Resultat handle Mail-Deskriptor

Siehe Verwandte Befehle, MailClose(),
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.

Als Mail-Typ (int1) können folgende Konstanten angegeben werden:

- MailSmtp (0) Unverschlüsselte Verbindung aufbauen (Standardport 25, falls nicht angegeben)
- MailSmtpTls (1) Verschlüsselte Verbindung mit STARTTLS aufbauen, wenn unterstützt (Standardport 25, falls nicht angegeben)
- MailSmtpS (2) Verschlüsselte Verbindung mit SMTPS aufbauen (Standardport 587, falls nicht angegeben)
- MailCreateEML (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 SMTPS-Verfahren versendet. Dabei wird beim Verbindungsaufbau 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 Verbindungsaufbau 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.

obj ->

MailData(int1,
alpha2[, alpha3[,
alpha4]]) : int



E-Mail definieren

obj Mail-Deskriptor

int1 Datentyp (siehe Text)

alpha2 Dateninhalt

alpha3 erweiterter Dateninhalt
(optional)

Objektreferenz /

alpha4 "Delivery Status
Notification"-Kommandos
(optional)

Resultat int Ergebnis



Siehe Verwandte Befehle,
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

obj ->
 MailClose(int1[, 
 var alpha2]) : int
 E-Mail versenden
 obj Mail-Deskriptor
 Verarbeitungsart
 _SmtplibSendNow Nachricht
 int1 versenden
 _SmtplibDiscard Nachricht
 verwerfen

 var Letzte Antwort des
 alpha2 Mailservers bei
 Fehlern (optional)
 Resultat int Ergebnis 
 Verwandte Befehle,
 Siehe MailOpen(),
 MailData(), Blog

Mit diesem Befehl wird die aufbereitete Mail (obj) entweder abgesendet (int1 = _SmtplibSendNow) oder verworfen (int1 = _SmtplibDiscard).

Wird beim Aufruf im optionalen Argument (alpha2) eine Variable zusammen mit der Option _SmtplibSendNow 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 (<u>_ErrOk</u>)	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	<i>Fehler beim Verbindungsaufbau</i>
23	Service nicht verfügbar
27	Sonstiger Fehler
30 - 39	<i>Fehler bei der Identifizierung</i>
31	Syntaxfehler
32	Befehl oder Parameter unbekannt
33	Service nicht verfügbar
37	Sonstiger Fehler
38	Fehler bei der Authentifizierung
40 - 49	<i>Fehler bei einer Empfangsadresse</i>

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	<i>Fehler bei der Datenübergabe</i>
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>DATA
```

Unterstützt der Mailserver kein ESMTP, muss statt EHLO HELO angegeben werden.

Mögliche Laufzeitfehler:

ErrHdlInvalid Deskriptor ungültig

Konstanten für E-Mail-Kopf
Konstanten für den E-Mail-Kopf
Siehe E-Mail-Befehle

- _SmtkBCC
- _SmtbCC
- _SmtbDiscard
- _SmtbFrom
- _SmtbPriority
- _SmtbReplyTo
- _SmtbSendNow
- _SmtbSubject
- _SmtbTo

_SmtkBCC
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.

_SmtpCC
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.

_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.

_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.

_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.

_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.

_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.

_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.

_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.

_SmtpXtension
Nicht standardisierter 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 Virensclannern 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 Express

Konstanten für E-Mail-Körper
Konstanten für den E-Mail-Körper
Siehe E-Mail-Befehle

- MailBuffer
- MailFile
- MailFileEML
- MailLine
- MimeTextHtml

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.

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 Nachrichteninhalte definiert, kann kein weiterer Inhalt oder Anhang angehängt werden.

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.

_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
- MimeOtherOP
- MimeRelated
- MimeTE 8B
- MimeTE B64
- MimeTE OP

_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.

_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.

_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.

_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.

_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.

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.

_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.

_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.

_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.

_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.

_MimeCS_UTF8
Unicode-Zeichensatz
Wert 240 /
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 umwandelnMail->MailData(_MailLine | _MimeTE_QP | _MimeCS_UTF8, StrCnv(StrCnv(
```


_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.

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.

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.

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).

MimeRelated
Eingebetteter Anhang
Wert 1.048.576 /
0x00100000

Verwandte
Siehe Befehle,
MailData()

Option bei MailData() - Anhang wird in den E-Mail-Body eingebettet.

_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.

MimeTE_B64
Base64 Kodierung
Wert 2 /
0x00000002

Verwandte

Siehe Befehle,
MailData()

Option bei MailData() - Kodierung nach Base64.

Mit dieser Option wird beim Befehl MailData() der Nachrichteninhalt nach Base64 kodiert.

_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 Nachrichteninhalte nach quoted-printable kodiert.

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().

```
... 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 OdbcErrSqlMessage 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

- FldAttributes
- ProcAdvertise



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 odbcc32.dll konnte nicht geladen werden.



obj -> OdbcClose()
Schließen von ODBC-Objekten

obj Deskriptor eines
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:

__ErrHdllInvalid Bei dem übergebenen Deskriptor handelt es sich nicht um ein gültiges ODBC-Objekt.

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 # tOdbcSourceList
```

Ermitteln aller System-Datenquellen nach Namen sortiert (ohne Berücksichtigung der Groß-/Kleinschreibung):

```
tOdbcApi # OdbcOpen();tOdbcSourceList # tOdbcApi->OdbcDataSources(_OdbcDataSourcesSystem | _OdbcDataSourcesUser)
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcApi-Objekt übergeben.

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

Siehe Verwandte Befehle, OdbcOpen(),
 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()
- OdbcCatalogCln()
- 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:

ErrHdlInvalid 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:

ErrHdllInvalid Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges OdbcApi-Objekt.

obj -> OdbcCatalogTbl([alpha1[,
alpha2[, alpha3[, alpha4]]]]) : 
handle

Tabelleninformationen ermitteln

obj Deskriptor eines
 ODBC-Verbindungs-Objekt

alpha1 Name der Tabelle

alpha2 Tabellen-Typ

alpha3 Name des Katalogs

alpha4 Name des Schemas

Resultat handle Deskriptor auf ein
 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:

ErrHdlInvalid Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges OdbcCon-Objekt.

obj ->
 OdbcCatalogCln([alpha1[,
 alpha2[, alpha3[, alpha4]]]]) : 
 handle

Spalteninformationen ermitteln
 obj Deskriptor auf ein
 ODBC-Verbindungs-Objekt
 alpha1 Name der Tabelle
 alpha2 Name der Spalte
 alpha3 Name des Katalogs
 alpha4 Name des Schemas

Resultat handle Deskriptor auf ein
 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 OdbcCln-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', '');tOdbcCatCln # tOdbcCon-
```

Mögliche Laufzeitfehler:

_ErrHdlInvalid Der in (obj) übergeben Deskriptor ist kein Deskriptor auf ein gültiges OdbcCon-Objekt.

obj ->
 OdbcPrepare(alpha1) : 
 handle
 SQL-Statement vorbereiten
 obj Deskriptor eines
 ODBC-Verbindungs-Objekts
 alpha1 SQL-Statement

Resultat handle Deskriptor auf ein
 ODBC-Statement-Objekt

Verwandte Befehle,
 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->OdbcFetch
```



obj -> OdbcParamSet(int1, var2) : int
Parameter für ODBC-Statement übergeben

obj Deskriptor des
 ODBC-Statement-Objekts

int1 Parameterposition

var2 Parameterwert

Resultat int Fehlerwert

Verwandte Befehle,

Siehe 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 vorbereitent0dbcStm # t0dbcCon->OdbcPrepare('INSERT INTO Customer (ID,NAME,DESCRIPTION
```

Mögliche Laufzeitfehler:

_ErrHdlInvalid

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.

obj -> OdbcParamAdd(int1[, int2[, logic3]])
: int



Parameter für ODBC-Statement definieren

obj Deskriptor eines
 ODBC-Statement-Objekts

int1 Datentyp

int2 maximale Länge der
 Zeichenkette (optional)

logic3 Länge der Zeichenkette
 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.

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. Genauere 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:

_ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.



obj -> OdbcExecute() : int;

Vorbereitetes ODBC-Statement ausführen

obj Deskriptor eines
ODBC-Statement-Objekts

Resultat int Fehlerwert 

Verwandte Befehle,

Siehe OdbcPrepare(),
 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:

_ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.

Kontakt

obj -> OdbcExecuteDirect(alpha1) : handle 
SQL-Statement vorbereiten und ausführen
obj Deskriptor eines ODBC-Verbindungs-Objekts
alpha1 SQL-Statement

Resultat handle Deskriptor eines 
ODBC-Statement-Objekts

Siehe Verwandte Befehle, OdbcConnect(),
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.



obj -> OdbcClmInfo(int1) : int
Informationen zu einer Spalte ermitteln

obj Deskriptor eines
 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:

_ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein OdbcStm-Objekt übergeben.



obj -> OdbcClnData(int1, var2) : int
Wert einer Spalte aus Ergebnismenge lesen

obj Deskriptor auf ein
 ODBC-Statement-Objekt

int1 Nummer der Spalte

var2 Variable zur Aufnahme
 des Werts

Resultat int Fehlerwert

Verwandte Befehle,
Siehe OdbcExecute(),
 OdbcExecuteDirect(),
 OdbcClnInfo()

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 OdbcResCountCln 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 OdbcClnInfo() 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:

<u>ErrHdlInvalid</u>	Im (obj) wurde kein gültiger Deskriptor auf ein <u>OdbcStm</u> -Objekt übergeben. Die angegebene Variable (var2) ist vom Typ <u>handle</u> , enthält aber keinen Deskriptor auf ein <u>Memory</u> -Objekt.
<u>ErrValueRange</u>	Die zu lesenden Daten sind größer als der vom <u>Memory</u> -Objekt bereitgestellte Speicher. Dieses Fehler tritt auch bei der Option <u>MemAutoSize</u> auf, wenn kein Speicher mehr angefordert werden kann.
<u>ErrMemExhausted</u>	Speicheranforderung fehlgeschlagen.

obj -> OdbcFetch([int1]) : int 
 Zeile der Ergebnismenge lesen
 Deskriptor auf ein
 ODBC-Statement-,
 obj -Tabellen- oder
 -Spalten-Objekt
 int1 reserviert, muss 0
 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 durchführentOdbcStm # tOdbcCon->OdbcExecuteDirect('INSERT INTO Customer (ID,NAME) VALU
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Im (obj) wurde kein gültiger Deskriptor auf ein zulässiges Objekt

Kontakt

übergeben.

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

Siehe Verwandte Befehle, ODBC-Schnittstelle,
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 gleichen 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

ProcAdvertise(alpha1, alpha2[,
logic3]) : logic
"Stored Procedure" bekannt geben



alpha1 Prozedur- und
Funktionsname
alpha2 Symbolischer Name
Funktionsparameter als
logic3 In/Out-Parameter
registrieren (optional)

Resultat logic Erfolg des Befehls

Siehe Verwandte Befehle,
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) angegebenen 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

- SckVolRead
- SckVolWrite

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 (_ErrHdlInvalid).



In der Beispiel-Datenbank "CodeLibrary" befindet sich ein Beispiel zur Kommunikation mit einem FTP-Server.

Mögliche Laufzeitfehler:

_ErrHdlInvalid 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
Nagle-Algorithmus ein

SckOptDontLinger Schaltet den
Linger-Modus aus

SckProxySOCKSv4 SOCKS-Proxyserver
Version 4

SckProxySOCKSv4a SOCKS-Proxyserver
Version 4a

SckProxySOCKSv5 SOCKS-Proxyserver
Version 5

SckTlsMax Verschlüsselung mit
maximalen
Sicherheitsanforderungen

SckTlsHigh Verschlüsselung mit
hohen
Sicherheitsanforderungen

int3

SckTlsMed Verschlüsselung mit
durchschnittlichen
Sicherheitsanforderungen

SckTlsLow Verschlüsselung mit
niedrigen
Sicherheitsanforderungen

SckNoTLSv1 Verschlüsselung ohne
Unterstützung von TLS 1.x

SckTlsSNI Verwendung von Server
Name Indication (SNI)
beim verschlüsselten
Verbindungsaufbau

SckOptVerify 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)

word6 Proxy-Portnummer /
Bind-Portnummer (optional)

alpha7 Proxy-Benutzer (optional)

alpha8 Proxy-Password (optional)

Resultat handle Deskriptor / Fehlerwert 

Siehe Verwandte Befehle, SckClose(),
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 kleine 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 Verbindungsaufbau erfolgt mit der Angabe

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 Verbindungsaufbau 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

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:

<u>ErrSckNoLib</u>	Das TCP/IP-Protokoll konnte nicht initialisiert werden
<u>ErrSckHostUnknown</u>	Der Name des Ziel-Hosts konnte nicht in eine IP-Adresse übersetzt werden
<u>ErrSckCreate</u>	Der Socket konnte nicht angelegt werden
<u>ErrSckConnectFailed</u>	Es konnte keine Verbindung zum Ziel-Host aufgebaut werden
<u>ErrSckBindFailed</u>	Es konnte keine Zuordnung zum angegebenen Port erfolgen
<u>ErrSckProxyUnknown</u>	Der Name des Proxys konnte nicht in eine IP-Adresse übersetzt werden
<u>ErrSckProxyRefused</u>	Der Proxy hat den Verbindungswunsch abgelehnt
<u>ErrSckProxyConnectFailed</u>	Es konnte keine Verbindung zum Proxy aufgebaut werden
<u>ErrSckProxyRead</u>	Bei der Kommunikation mit dem Proxy ist ein Lesefehler aufgetreten
<u>ErrSckProxyWrite</u>	Bei der Kommunikation mit dem Proxy ist ein Schreibfehler aufgetreten
<u>ErrSckTlsConnect</u>	Der verbundene Server unterstützt keine der angegebenen verschlüsselten Verbindungen oder der TLS-Handshake ist fehlgeschlagen.
<u>ErrSckTlsCertificateVerify</u>	Fehler bei der Überprüfung des serverseitigen Zertifikats



obj -> SckInfo(int1[, int2]) : alpha
 Socket-Informationen abfragen/setzen

obj Socket-Deskriptor

int1 Informationstyp
 (siehe Text)

int2 optionaler Wert
 (optional)

Resultat alpha Informationen

Siehe Verwandte Befehle

Mit diesem Befehl können Informationen über die Socket-Verbindung abgefragt oder gesetzt werden. In (obj) wird der Socket-Deskriptor übergeben.



In der Beispiel-Datenbank "CodeLibrary" befindet sich ein Beispiel zur Kommunikation mit einem FTP-Server.

Folgende Informationen können ermittelt werden:

Option (int1)	zurückgegebene Information
<u>SckAddrLocal</u>	lokale IP-Adresse
<u>SckAddrPeer</u>	IP-Adresse des Zielhosts
<u>SckCertificateCN</u>	"Common Name" des serverseitigen Zertifikats
<u>SckHostName</u>	lokaler Hostname
<u>SckPortLocal</u>	lokaler IP-Port
<u>SckPortPeer</u>	IP-Port des Zielhosts
<u>SckVersionTLS</u>	verwendete TLS-Version bei verschlüsselter Verbindung
<u>SckVol</u>	übertragene Datenvolumen in Bytes
<u>SckVolRead</u>	empfangene Datenvolumen in Bytes
<u>SckVolWrite</u>	gesendete Datenvolumen in Bytes
<u>SckReadyRead</u>	Ermitteln ob Daten zum Lesen vorhanden
<u>SckTimeout</u>	Ermitteln / Setzen des Timeout
<u>SckKeepAlive</u>	Ermitteln / Setzen der Keep-Alive-Zeit
<u>SckDefaultKeepAlive</u>	Setzen der Keep-Alive-Zeit auf den eingestellten Vorgabewert (siehe <u>SOA-Service - Konfigurationsdatei</u>)

Wird mit SckVersionTLS die TLS-Version einer verschlüsselten Verbindung ermittelt, wird Version gefolgt von einem Leerzeichen und der verwendeten Cipher-Suite 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

Wird bei einer verschlüsselten Verbindung das Server-Zertifikat überprüft (siehe SckOptVerify) muss anschließend geprüft werden, ob das Zertifikat auch für den verbundenen Server ausgestellt wurde. Mit tSck->SckInfo(_SckCertificateCN) wird der "Common Name" aus dem Zertifikat ermittelt und kann mit dem Hostnamen verglichen werden. Da der "Common Name" Wildcards beinhalten kann, muss ein

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)

int3 SckOptDelay Schaltet den
Nagle-Algorithmus
ein

SckOptDontLinger Schaltet den
Linger-Modus aus

handle4 Deskriptor des Frame-Objektes
(optional)

Resultat handle Deskriptor / Fehlerwert

Siehe Verwandte Befehle, SckConnect(),
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 neuer 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.

obj -> SckRead(int1, ,
var2[, int3]) : int
Daten vom Socket lesen
obj Socket-Deskriptor
 Optionen
 SckLine komplette
 Zeile lesen
int1 SckReadMax Lesen bis
 maximale
 Länge / Lesen
 aller
 empfangener
 Zeichen
var2 Feld, Variable oder Array
int3 Anzahl zu lesender Bytes
 (optional)
Resultat int Anzahl der gelesenen
 Bytes oder Fehlerwert
Siehe Verwandte Befehle,
 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

(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
<u>ErrTimeout</u>	Beim Lesen oder Schreiben des Sockets ist ein Timeout aufgetreten

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.



obj -> SckStartTls(int1[, alpha2]) : int

TLS-Verschlüsselung für Socket-Verbindung einleiten

obj Socket-Deskriptor

Optionen

	<u>SckTlsMax</u>	Verschlüsselung mit maximalen Sicherheitsanforderungen
	<u>SckTlsHigh</u>	Verschlüsselung mit hohen Sicherheitsanforderungen
	<u>SckTlsMed</u>	Verschlüsselung mit durchschnittlichen Sicherheitsanforderungen
int1	<u>SckTlsLow</u>	Verschlüsselung mit niedrigen Sicherheitsanforderungen
	<u>SckNoTLSv1</u>	Verschlüsselung ohne Unterstützung von TLS 1.x
	<u>SckTlsSNI</u>	Verwendung von Server Name Indication (SNI) beim verschlüsselten Verbindungsaufbau
	<u>SckOptVerify</u>	Überprüfung des serverseitigen Zertifikats
alpha2	Hostname bei Verwendung von SNI (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 ausgeben sub DbgHeader( aRsp                      : handle;) local {
```

Mögliche Laufzeitfehler:

ErrHdlInvalid Der angegebene Deskriptor (obj) ist ungültig oder verweist nicht auf eine Socket-Verbindung.

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

int3 Anzahl zu schreibender Bytes
 (optional)

Resultat int Anzahl der geschriebenen
 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

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

Vom Memory-Objekt auf den Socket schreiben

obj Socket-Deskriptor

Optionen

int1 SckBuffered Daten puffern

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.

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

Konstanten für Socket-Befehle

Siehe [Alle Befehle](#)

_SckBuffered	Daten puffern
_SckLine	CR/LF verarbeiten
_SckReadMax	Lesen bis maximale Länge / Lesen aller empfangener Zeichen
_SckAddrLocal	Die lokale IP-Adresse
_SckAddrPeer	Die IP-Adresse des Zielhosts
_SckCertificateCN	"Common Name" des Zertifikats
_SckDefaultKeepAlive	Keep-Alive-Zeit auf den Vorgabewert setzen
_SckHostName	Der lokale Hostname
_SckPortLocal	Der lokale IP-Port
_SckPortPeer	Der IP-Port des Zielhosts
_SckVol	Das übertragene Datenvolumen in Bytes
_SckVolRead	Das empfangene Datenvolumen in Bytes
_SckVolWrite	Das gesendete Datenvolumen in Bytes
_SckReadyRead	Ermitteln ob Daten zum Lesen vorhanden sind
_SckOptDelay	Schaltet den Nagle-Algorithmus ein
_SckOptDontLinger	Schaltet den Linger-Modus aus
_SckProxySOCKSv4	Version 4 des SOCKS-Proxyserver verwenden
_SckProxySOCKSv4a	Version 4a des SOCKS-Proxyserver verwenden
_SckProxySOCKSv5	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.

_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.

_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.

_SckNoTLsv1
Verschlüsselung ohne Unterstützung von TLS 1.x
Wert 8.388.608 /
0x00800000

Siehe SckConnect(),
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.

_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.

_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 Verbindungsaufbau werden die TCP-Datensegmente beim Sender so lange zwischengepuffert, bis ein volles Datensegment übertragen werden kann.

_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.

SckOptVerify

Überprüfung des serverseitigen Zertifikats

Wert 16.777.216 /
0x01000000

Siehe SckConnect(),
SckStartTls()

Option bei den Anweisungen SckConnect() und SckStartTls() - Bei Verwendung einer verschlüsselte 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.

_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.

_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.

_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.

_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.

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	Verständnis
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any	AEAD (256-Bit)
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any	CHACHA20-POLY1305 (256-Bit)
TLS_AES_128_GCM_SHA256	TLSv1.3	any	any	AEAD (128-Bit)
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AEAD (256-Bit)
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AEAD (256-Bit)
ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	ECDH	RSA	CHACHA20-POLY1305 (256-Bit)
DHE-RSA-CHACHA20-POLY1305	TLSv1.2	DH	RSA	CHACHA20-POLY1305 (256-Bit)

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	AF	(2)
DHE-RSA-ARIA256-GCM-SHA384	TLSv1.2	DH	RSA	AF	(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	AF	(1)
DHE-RSA-ARIA128-GCM-SHA256	TLSv1.2	DH	RSA	AF	(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	

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

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	Verfahren
TLS_AES_256_GCM_SHA384	TLSv1.3	any	any	AEAD (256 Bit)
TLS_CHACHA20_POLY1305_SHA256	TLSv1.3	any	any	CHACHA20-POLY1305 (256 Bit)
TLS_AES_128_GCM_SHA256	TLSv1.3	any	any	AEAD (128 Bit)
ECDHE-RSA-AES256-GCM-SHA384	TLSv1.2	ECDH	RSA	AEAD (256 Bit)
DHE-RSA-AES256-GCM-SHA384	TLSv1.2	DH	RSA	AEAD (256 Bit)
ECDHE-RSA-CHACHA20-POLY1305	TLSv1.2	ECDH	RSA	CHACHA20-POLY1305 (256 Bit)

	Kontakt			
DHE-RSA-CHACHA20-POLY1305	TLSv1.2	DH	RSA	
DHE-RSA-AES256-CCM8	TLSv1.2	DH	RSA	
DHE-RSA-AES256-CCM	TLSv1.2	DH	RSA	
ECDHE-ARIA256-GCM-SHA384	TLSv1.2	ECDH	RSA	
DHE-RSA-ARIA256-GCM-SHA384	TLSv1.2	DH	RSA	
ECDHE-RSA-AES256-SHA384	TLSv1.2	ECDH	RSA	
DHE-RSA-AES256-SHA256	TLSv1.2	DH	RSA	
ECDHE-RSA-CAMELLIA256-SHA384	TLSv1.2	ECDH	RSA	
DHE-RSA-CAMELLIA256-SHA256	TLSv1.2	DH	RSA	
ECDHE-RSA-AES256-SHA	TLSv1.2	ECDH	RSA	
DHE-RSA-AES256-SHA	SSLv3	DH	RSA	
DHE-RSA-CAMELLIA256-SHA	SSLv3	DH	RSA	

_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

_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 Verbindungsaufbau Server Name Indication (SNI) verwendet wird.

Die Konstante muss mit einer der Konstanten _SckTlsLow, _SckTlsMed, _SckTlsHigh oder _SckTlsMax kombiniert werden, damit sie sich auswirkt.

`_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

SckVol

Übertragene Datenvolumen in Bytes

Wert 5

Siehe SckInfo()

Option bei SckInfo() - Das übertragene Datenvolumen in Bytes ermitteln.

_SckVolRead

Empfangene Datenvolumen in Bytes

Wert 6

Siehe SckInfo()

Option bei SckInfo() - Das empfangene Datenvolumen in Bytes ermitteln.

_SckVolWrite

Gesendete Datenvolumen in Bytes

Wert 7

Siehe SckInfo()

Option bei SckInfo() - Das gesendete Datenvolumen in Bytes ermitteln.

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 versteht 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 dass 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.

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.

MsxOpen(int1, handle2) 

: handle

Nachrichtenkanal öffnen

Modus

MsxSocket Kommunikation
über
Socket-Verbindung

MsxFile Kommunikation
über externe Datei

int1 MsxThread Kommunikation mit
einem als Thread
gestarteten Job

MsxProcess Kommunikation mit
einem als Prozess
gestarteten Job

MsxRead Lesender Zugriff

MsxWrite Schreibender
Zugriff

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 Verbindungsaufbau 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 versendenSck # SckConnect('10.1.1.16', 1250);if (tSck > 0){ tMsx # Msx
```

Folgende Laufzeitfehler sind möglich:

<u>_ErrHdlInvalid</u>	Der übergebene Socket- oder Dateideskriptor ist ungültig.
<u>_ErrValueInvalid</u>	Es wurde in (int1) eine ungültige Konstantenkombination übergeben.

obj -> MsxRead(int1, ) : 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
 _ErrEndOfData Keine weiteren
 Nachrichten (bei
 _MsxFile) 
 Resultat int _ErrData Allgemeiner 
 Datenfehler
 _ErrSck... Socket-Fehler
 _ErrFsi... Fehler externe
 Dateioperationen

Siehe Verwandte Befehle, MsxOpen(),
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 _ErrEndOfData.

- **_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(_MsxMessag
```

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.

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

int


Nachrichtenkanal in Memory-Objekt lesen

obj Deskriptor eines
 Nachrichtenkanals

handle1 Deskriptor eines
 Memory-Objekts

int2 Zielposition im
 Memory-Objekt

int3 Anzahl der zu
 lesenden Bytes

Resultat int Fehlerwert 

Siehe Verwandte Befehle,
 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.

obj -> MsxWrite(int1, var2[, 
int3]) : int

Nachrichteninhalt schreiben

obj	Deskriptor eines Nachrichtenchannels Nachrichtensbereich <u>MsxMessage</u> Nachrichten-ID
int1	<u>MsxItem</u> Bereichs-ID <u>MsxDat</u> Datenbereich <u>MsxEnd</u> Nachrichtenpuffer abschließen
var2	Variable oder Konstante mit dem zu schreibendem Inhalt
int3	Länge des Inhalts (optional)
Resultat	<u>int</u> Fehlerwert 
Siehe	<u>Verwandte Befehle</u> , <u>MsxOpen()</u> , <u>MsxRead()</u> , <u>Socket-Fehler</u> , <u>Fehler externe</u> <u>Dateioperationen</u>

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.

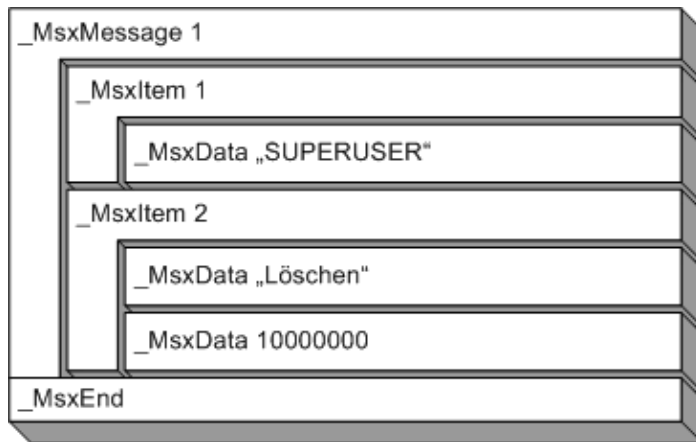
- **MsxDat**

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. MsxDat 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.



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.

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

_MsxDat
Datenbereich lesen/schreiben
Wert 3 / 0x03

Verwandte
Befehle,
Siehe MsxRead(),
MsxWrite()

Option der Befehle MsxRead() und MsxWrite() - Es wird ein Datenfeld gelesen, beziehungsweise geschrieben.

_MsxEnd
Nachricht abschließen
Wert 4 / 0x04

Verwandte

Siehe Befehle,
MsxWrite(),
MsxRead()

Option der Befehle MsxWrite() und MsxRead() - Das Nachrichtenende wird gelesen, beziehungsweise geschrieben.

_MsxFile

Kommunikation über externe Datei

Wert 1 / 0x01

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Die Kommunikation findet über eine externe Datei statt.

_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.

_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.

_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.

_MsxSocket
Kommunikation über Socket
Wert 2 / 0x02

Verwandte

Siehe Befehle,

MsxOpen()

Option des Befehls MsxOpen() - Die Kommunikation findet über eine Socket-Verbindung statt.

_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.

_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	
	<u>TapiCallOpAnswer</u>	Gespräch annehmen
	<u>TapiCallOpDrop</u>	Gespräch trennen
	<u>TapiCallOpHold</u>	Gespräch halten
	<u>TapiCallOpOrigin</u>	Ursprung des Gesprächs ermitteln
int1	<u>TapiCallOpState</u>	Gesprächsstatus ermitteln
	<u>TapiCallOpUnhold</u>	Gespräch wieder aufnehmen
	<u>TapiCallOpSetOwner</u>	Applikation wird Eigentümer des Anrufs
	<u>TapiCallOpIsOwner</u>	Überprüfung der Eigentümerschaft

int2 wird nicht ausgewertet

Resultat 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:

<u>TapiCallOpAnswer</u>	Gespräch annehmen
<u>TapiCallOpDrop</u>	Gespräch trennen
<u>TapiCallOpHold</u>	Gespräch halten
<u>TapiCallOpOrigin</u>	Ursprung des Gesprächs ermitteln
<u>TapiCallOpState</u>	Gesprächsstatus ermitteln
<u>TapiCallOpUnhold</u>	Gespräch wieder aufnehmen
<u>TapiCallOpSetOwner</u>	Applikation wird Eigentümer des Anrufs
<u>TapiCallOpIsOwner</u>	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.



obj -> TapiClose()

TAPI-Gerätetreiberliste schließen

obj Liste der
 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().

obj ->

TapiConference() : 

int

Konferenz initiieren

obj Call-ID

Resultat int ConsultCall ID oder
Fehlerwert

Verwandte Befehle,

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.

obj ->

TapiConferenceComm()

: 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 _TapiCallStateOnHoldPendTransfer 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.

obj -> `TapiConferenceDial(alpha1[,` 
`int2, [int3]]) : int`

Dritten Konferenzteilnehmer anrufen

obj ConsultCall-ID

alpha1 Wählstring

Optionen (optional)

0 Keine Option

int2 TapiAsyncDial Asynchrones
 Wählen

TapiRawDial Wählstring
 unverändert
 benutzen

int3 Timeout beim synchronen
 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 abubrechen. 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 Deskriptor des
 TAPI-Gerätetreibers

alpha1 Wählstring
 Optionen (optional)

 0 Keine Option

TapiAsyncDial Asynchrones
 Wählen

int2 TapiReturnCallID Asynchrones
 Wählen mit
 Rückgabe
 der Call-ID

TapiRawDial Wählstring
 unverändert
 benutzen

int3 Timeout beim synchronen
 Wählen in Millisekunden
 (optional)

Resultat int ErrOk, resultierende
 Call-ID oder Fehlerwert

Siehe Verwandte Befehle,
TapiOpen(), Blog

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

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

int2 TapiAsyncDial Asynchrones Wählen

TapiRawDial Wählstring unverändert benutzen

int3 Timeout beim synchronen Wählen (optional)

Resultat int Fehlerwert

Siehe Verwandte Befehle,
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.



obj -> TapiInfo(int1, var var2) : int
Informationen zu einem Anruf ermitteln

obj	Call-ID oder Deskriptor des TAPI-Gerätetreibers	
	Informationstyp	
	<u>TapiInfoCallerID</u>	Nummer des Anrufers
	<u>TapiInfoCallerIDName</u>	Name des Anrufers
	<u>TapiInfoCalledID</u>	Nummer des Angerufenen
	<u>TapiInfoCalledIDName</u>	Name des Angerufenen
int1	<u>TapiInfoConnectedID</u>	Nummer des verbundenen Teilnehmers
	<u>TapiInfoConnectedIDName</u>	Name des verbundenen Teilnehmers
	<u>TapiInfoStatus</u>	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**

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

<u>ErrFldType</u>	In (var2) wird der falsche Variablentyp übergeben.
<u>ErrValueInvalid</u>	In (int1) wurde ein falscher Informationstyp angegeben.
<u>ErrHdlInvalid</u>	Bei dem übergebenen Deskriptor (bei <u>TapiInfoStatus</u>) handelt es sich nicht um ein Tapi-Gerät oder es ist kein <u>TapiListen()</u> auf dem Tapi-Gerät aktiv.

obj -> TapiListen(logic1,
handle2) : int



Eingehende Anrufe überwachen

obj Deskriptor des
 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 Geräte-Treibers 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

TapiOpen() : handle



TAPI-Gerätetreiber ermitteln

Resultat handle Deskriptor einer
Cte-Liste



Siehe Cte-Befehle, Verwandte Befehle,
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 ermittelt
tTapiHdl # TapiOpen();
if (tTapiHdl > 0){
    tTapiDevice # t
```


obj ->

TapiPickup(alpha1 

: int

Anruf heranzuholen

obj Deskriptor des
 TAPI-Gerätetreibers

alpha1 Zielapparat

Resultat int CallID oder
 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 herangezogen 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

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](#)
- [_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

WseArg(int1, alpha2[,
int3]) : alpha



Argumente ermitteln

Parametertyp

WseArgName Parametername
ermitteln

int1 WseArgReq Parameterwert
ermitteln

WseArgCall Übergabeparameter
ermitteln

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

<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 ParametertCounter # 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 ParametertCounter # 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 \equiv) 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 "".

WseInfo(int1[, alpha2]) :



alpha

Web-Informationen ermitteln

int1 Zu ermittelnde
 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

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

	<u>WseRetHeader</u>	Es wird nur der HTTP-Header übertragen
	<u>WseRetString</u>	In (var4) wird eine Zeichenkette angegeben
int1	<u>WseRetText</u>	Ein bestehender Text wird übertragen
	<u>WseRetFile</u>	Eine externe Datei wird übergeben
	<u>WseRetExpires</u>	Verfallsdatum für die übertragenen Daten definieren
int2	Optionen zur Verarbeitung der Rückgabedaten	
	<u>WseNoParse</u>	Die zurückgegebenen Daten werden nicht durch den Client interpretiert
	<u>WseSendAfter</u>	Die zurückgegebenen Daten werden erst nach dem Beenden der Prozedur an den Web-Server übergeben
	<u>WseTerm</u>	Die Session wird nach der Rückgabe beendet
	<u>WseCnvISO</u>	Der Rückgabewert wird in den Zeichensatz ISO 8859-1 (Latin-1) umgewandelt
	<u>WseCnvHTML</u>	Sonderzeichen werden gewandelt
	<u>WseCnvCtrl</u>	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 Verwandte Befehle

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:

- **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 &#lt;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 registrierter MIME-Typen kann über die Homepage der IANA (<http://www.iana.org/>) abgerufen werden. Nicht standardisierte MIME-Subtypen

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
<u>WseRetHeader</u>	Felder und Werte des HTTP-Response-Headers
<u>WseRetString</u>	Zeichenkette
<u>WseRetText</u>	Deskriptor des Textpuffers
<u>WseRetFile</u>	Pfad und Dateiname
<u>WseRetExpires</u>	Verfallsdatum der übertragenen Daten vom Typ <u>caltime</u>

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.



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 : ...
```

WseStrCnv(int1, alpha2[,
alpha3]) : alpha



Zeichenkette konvertieren

Art der Umwandlung

int1	<u>_WseCnvISO</u>	Der Rückgabewert wird in den Zeichensatz ISO 8859-1 (Latin-1) umgewandelt
	<u>_WseCnvHTML</u>	Sonderzeichen werden umgewandelt
	<u>_WseCnvCtrl</u>	Steuerzeichen im HTML-Text werden umgewandelt
	<u>_WseCnvArg</u>	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:

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.

_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.

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.

_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.

_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 &

_WseCnvHTML

Sonderzeichen werden umgewandelt

Wert 2 / 0x02

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). Sonderzeichen werden in &#lt;Nummer>; gewandelt.

_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.

_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.

_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 ([...]).

_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.

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.

_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.

_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.

_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="hidden"
```

CONZEPT 16-Prozedur:

```
local{ tReqData : alpha(4096);}...tReqData # WseInfo(_WseInfoReqData);...
```

_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.

_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.

_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.

_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.

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.

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.

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.

_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.

WseRetFile
Übergabe einer externen Datei
Wert 3

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). In (var4) steht der Pfad und der Name der Datei.

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' ...

_WseRetString
Angabe einer Zeichenkette
Wert 1

Verwandte

Siehe Befehle,

WseReturn()

Option bei WseReturn(). In (var4) wird eine Zeichenkette angegeben.

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.

_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.

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.

_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.

_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.

_WseUserProc
Status des Web-Benutzers
Wert 1

Verwandte

Siehe Befehle,

WseStatus()

Option bei WseStatus(). Die Request-Prozedur wurde erneut aufgerufen.

_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



obj -> PrtPropGet(int1, var2) : logic

Auslesen einer Eigenschaft eines Druckobjektes

obj Objekt

int1 Konstante der
 Eigenschaft

var2 Wert der Eigenschaft

Resultat logic Funktion erfolgreich

Verwandte Befehle,

Siehe 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 Objektitels$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;
```



obj -> PrtPropSet(int1, var2[, int3]) : logic
Auslesen einer Eigenschaft eines Druckobjektes

obj Objekt
int1 Konstante der Eigenschaft
var2 Zu setzender Wert
int3 Zusätzliche Information
Resultat logic Funktion erfolgreich
 Verwandte Befehle,
Siehe PrtPropGet(),
 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 Objektitels auf "Neu"$Text->PrtPropSet(_PrtPropCaption, 'Neu');
```



Alternativ kann die Eigenschaft auch wie folgt gesetzt werden:

Beispiel:

```
// Setzen des Objektitels 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.

PrtJobOpen(alpha1[, alpha2[,
int3[, int4[, alpha5[, int6]]]])



: handle

Druckjob anlegen

Druck-Dokument Name oder

alpha1 PrtDocDinA4 Din A4 Seitenformat

PrtDocDinA5 Din A5 Seitenformat

alpha2 Externer Dateiname

Optionen

PrtJobOpenWrite Druckjob zum
Schreiben öffnen

PrtJobOpenRead Druckjob zum
Lesen öffnen

PrtJobOpenTemp Temporären
Druckjob öffnen

int3 PrtJobOpenSort Visuelle
Reihenfolge der
Druckobjekte

PrtJobOpenEmbedImages Bilder einbetten

PrtJobOpenVerbose Druckjob mit
zusätzlichen
Informationen
anlegen

Objekttyp

int4 PrtTypePrintDoc PrintDoc-Objekt
drucken

PrtTypePrintDocRecord PrintDocRecord-Objekt
drucken

alpha5 Krypt-Key

Unicode-Unterstützung

PrtConvNone Das Objekt laden, wie es
gespeichert wurde.

int6 PrtConvUnicode Unicode-Unterstützung
aktiviert

PrtConvAnsi keine
Unicode-Unterstützung

Resultat handle Objekt



Siehe Verwandte Befehle, PrtJobWrite(),
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

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**

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 | _PrtJob0
```

• 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,

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.

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

Siehe Verwandte Befehle,
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 SeitetAddSize # tPage->ppBoundAdd;// Maximal zu Verfügung stehend

obj -> PrtJobClose(int1[, ,
handle2]) : int

Druckjob schließen

obj Printjob-Objekt

Optionen (optional)

<u>PrtJobPreview</u>	Vorschau anzeigen
<u>PrtJobPreviewValidate</u>	Vorschau anzeigen und prüfen, ob der Druck-Dialog mit OK verlassen wurde
<u>PrtJobPrint</u>	sofort drucken
<u>PrtJobPdf</u>	als PDF-Datei drucken
<u>PrtJobTif</u>	als MultiPage-Tiff-Datei drucken
<u>PrtJobEmf</u>	als EMF-Datei(en) drucken
<u>PrtJobXml</u>	als XML-Datei drucken
<u>PrtJobHidden</u>	Fortschrittsanzeige während der Druckaufbereitung unterdrücken
<u>PrtJobCancel</u>	Druckaufbereitung abbrechen

int1

handle2 PrtDevice-Objekt (optional)

Resultat int Fehlerwert

Siehe Verwandte Befehle, PrtJobOpen(),
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:

- **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 fortgefahren. 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 dass 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:

<u>ErrOk</u>	Die Verarbeitung des Druckjobs war erfolgreich.
<u>ErrRange</u>	Keine zu druckende Seite oder Startseite größer Endeseite.
<u>ErrSystem</u>	Beim Drucken ist ein Fehler aufgetreten, der nicht weiter spezifiziert werden kann.
<u>PrtJobPreviewPrintClicked</u>	Die Option <u>PrtJobPreviewValidate</u> wurde angegeben und im Druck-Dialog wurde auf OK geklickt.
<u>ErrPrtPaperFormat</u>	Das Papierformat des <u>PrintDoc</u> -Objektes konnte auf der angegebenen <u>PrintDevice</u> nicht ausgegeben werden.
<u>ErrPdfPageAppend</u>	Es konnte keine Seite an das PDF-Dokument angehängt werden.
<u>ErrPdfInsertMetaFile</u>	Der Seite des PDF-Dokuments konnte kein Inhalt hinzugefügt werden.
<u>ErrPdfNotPdfA</u>	Es sollte ein PDF/A-Dokument erstellt werden. Dieser Fehler wird zurückgegeben, wenn nicht alle Informationen für ein PDF/A-Dokument geschrieben werden konnten.
<u>ErrFsi...</u>	Beim Erstellen der externen Datei ist ein Problem aufgetreten. Die entsprechenden Rückgabewerte sind im Abschnitt <u>Externe Dateioperationen</u> erläutert.

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)	
	<u>PrtConvNone</u>	Das Objekt laden, wie es gespeichert wurde.
int3	<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 Objektname (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

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

Siehe Befehle,
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
```

obj -> PrtAdd(handle1[, int2[, int3[,
int4]]) : int



PrintForm zu einem PrintJob hinzufügen

obj Seite des PrintJob-Objektes

handle1 PrintForm-Objekt

Optionen

PrtAddPageBreak automatischer
Seitenumbruch
PrtAddTop anderes
int2 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.

Beispiel:

```
// PrintForm an die Seite anhängentPage->PrtAdd(tPrintForm);// PrintForm an diese oder die nächste
```

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)
 PrtAddPageBreak automatischer
 Seitenumbruch
int2 PrtAddTop anderes Druckobjekt
 überdrucken
 PrtAddRelative relative Angabe der
 Druckposition
int3 Abstand vom Seitenanfang (optional)
int4 Abstand vom linken Seitenrand (optional)
 Unicode-Unterstützung (optional)
 PrtConvNone Das Objekt laden, wie es
 gespeichert wurde.
int5 PrtConvUnicode Unicode-Unterstützung
 aktiviert
 PrtConvAnsi keine
 Unicode-Unterstützung
 Objekttyp (optional)
int6 PrtTypePrintForm PrintForm-Objekt
 PrtTypePrintFormList 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**

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
```

obj -> PrtInfo(int1[, int2[,
int3]]) : int



Objektinformationen ermitteln

obj Objekt

int1 Optionen (siehe Text)

int2 Count (Optional)

int3 Typ (Optional)

Resultat int Gewünschte Information

Siehe [Verwandte Befehle](#),
[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(_PrtLis
// Alle Druckauflösungen des Standarddruckers ermittelntDevice # PrtDeviceOpen();if (tDevice > 0)
```


obj -> PrtInfoStr(int1[, int2]) 

: alpha

Ermitteln von Informationen

obj Objekt

Mode

int1 PrtInfoBinName Name des
Druckerschachts

PrtInfoPaperName Name des
Papierformats

int2 Index

Resultat alpha Gewünschte 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



obj -> PrtSearch(alpha1) : handle

Suchen eines Druck-Objekts über den Namen

obj Startobjekt der Suche

alpha1 Name des zu suchenden
 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
```

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

PrtRtfSearchCase Groß-/Kleinschreibung
beachten

int2 PrtRtfSearchWord Nur ganze Wörter
suchen

PrtRtfSearchReplace Suchbegriff durch
(alpha4) ersetzen

PrtRtfSearchDelete Bereich in (range3)
entfernen

range3 Suchbereich (optional)

alpha4 Ersetzungstext (optional)

var alpha5 Auf Suchtext folgende Zeichenkette
(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.

- **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

PrtUnit(int1, int2) : 

float

Umwandeln in Einheit

int1 Wert in logischen Einheiten

int2 Zieleinheit der
Umwandlung

Resultat float Umgewandelte Einheit

Siehe Verwandte Befehle,
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, _PrtU
```

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 Einheit  
tLeftMargin # PrtUnitLog(tLeftMargin, _PrtUnitCentimetres);
```

PrtDeviceOpen([alpha1, 
int2]) : handle

Drucker-Device öffnen

alpha1 Druckername

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
```

obj -> PrtDeviceClose() : 
logic

Drucker-Device schließen

obj Drucker-Device
 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->PrtDeviceClose()
```


Kontakt

obj -> PrtDeviceRefresh() : int
Ermitteln der Device-Informationen
obj Device-Objekt
Resultat int Status



Siehe Verwandte Befehle,
PrtPrinterRefresh()

Mit diesem Befehl werden alle Eigenschaften des in (obj) übergebenen PrintDevice-Objektes neu eingelesen. Der Befehl gibt _ErrGeneric zurück, falls beim Ermitteln der Eigenschaften des PrintDevice ein Fehler aufgetreten ist. Im Erfolgsfall wird _ErrOk zurückgegeben

Beispiel:

```
if (tPrintDevice->PrtDeviceRefresh() = _ErrOk){ // Eigenschaften des PrintDevice wurden erfolgre
```

Kontakt

obj -> PrtPrinterRefresh() : int



Aktualisieren der Drucker-Eigenschaften

obj Druckerobjekt-Deskriptor

Resultat int Status

Siehe Verwandte Befehle,
 PrtDeviceRefresh()

Eine Liste der Systemdrucker kann über die Eigenschaft PrinterList des _App-Objekts ermittelt werden. Der Deskriptor der Liste kann an PrtSearch() oder PrtInfo() übergeben werden, um den Deskriptor eines bestimmten Druckers zu ermitteln.

Zu Beginn sind nur die Eigenschaften Name und Comment lesbar. Zum Ermitteln der restlichen Eigenschaften des Druckerobjekts dient der Befehl PrtPrinterRefresh.

Beispiel:

```
tPrinterList # _App->PrinterList;tPrinter # tPrinterList->PrtInfo(_PrtFirst);if (tPrinter->PrtPri
```

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

obj -> PpcMakeAcrobatPdf(alpha1[, alpha2]) 

: int

PDF-Dokument mit Adobe Distiller erstellen

obj PpcObject-Deskriptor

alpha1 Pfad- und Dateiname
der PDF-Datei

alpha2 Einstellungen des
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 angegeben. 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

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 angegeben. 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.

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 angegeben. 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.

<u>ErrOk</u>	PDF-Erstellung erfolgreich
<u>ErrPpcDriver</u>	Falscher Druckertreiber (zum Beispiel TIFF)
<u>ErrPpcGhostscript</u>	Fehler bei der PDF-Erstellung

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,

Siehe PpcMakePreviewJpg(),
 PpcMakePreviewPng(),
 PpcMakePreviewTif()

Der Befehl erzeugt einzelne Preview-Dateien ausgewählter Seiten der Originaldatei im Windows Bitmap-Format.

In (obj) wird der Deskriptor des PpcObject angegeben. 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'

'<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.

obj -> PpcMakePreviewJpg(alpha1,
point2[, int3]) : int
Preview-Dateien im JPEG-Format erstellen



obj PpcObject-Deskriptor
Pfad- und Dateiname

alpha1 der erzeugten
Datei(en)

point2 Breite und Höhe des
Bildes

int3 JPEG-Qualität in
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 angegeben. 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'

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.

obj -> PpcMakePreviewPng(alpha1,
point2) : int



Preview-Dateien im PNG-Format erstellen

obj PpcObject-Deskriptor

Pfad- und Dateiname

alpha1 der erzeugten

Datei(en)

point2 Breite und Höhe des

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 angegeben. 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('pvw_%d_%d.png', PointMake(150,
```

Es werden dann folgende Dateien erzeugt: pvw_1_1.png, pvw_3_3.png, pvw_5_5.png, ...

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' 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.

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.

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,

Siehe PpcMakePreviewBmp(),
 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 angegeben. 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 Dateiname 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'

'<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.

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 angegeben. 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.



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->PrtDeviceC
```



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.

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](#)

- ◆ 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**

- ◆ ErrDbAreaInUse
- ◆ ErrDbAreaLocked
- ◆ ErrDbAreaLockedAdmin
- ◆ ErrDbAreaLockedDown
- ◆ ErrDbAreaLockedNoStandbyOpen
- ◆ ErrDbAreaLockedOpen
- ◆ ErrDbAreaLockedOperation
- ◆ ErrDbAreaLockedRollback
- ◆ ErrDbAreaLockedStandby
- ◆ ErrDbAreaOpen

- ◆ ErrDbAreaOperationDenied
- ◆ ErrDbAreaPassword
- ◆ ErrDbAreaRollback
- ◆ ErrDbAreaStandby
- ◆ ErrDbAreaType
- ◆ ErrDbComm
- ◆ ErrDbNoArea
- ◆ ErrDbNoServer
- ◆ ErrDbServerRelease
- ◆ ErrDbServerStart
- ◆ ErrDbUserInvalid
- ◆ ErrDbUserLimit
- ◆ ErrDbUserSelf

• **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

- **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

- **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

- ◆ 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

Siehe Befehle, try,
Laufzeitfehler
finden (Blog)

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:ErrHandler');
```

Bei einem Fehler wird die Funktion ErrHandler 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
- ErrMsg - 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 Debuggerverbindung erfolgt jedoch immer ein 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 ErrHandler{ ...}// odersub ErrHar
```

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.



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 ErrOutOfMemory
 -51 ErrRange
 -52 ErrType
 -53 ErrSystem
 -55 ErrInProgress
 -56 ErrNameInvalid
 -57 ErrAccessMode
 -58 ErrExists
 -59 ErrLocked
 -60 ErrLimitExceeded
 -61 ErrUnerasable
 -62 ErrUnchangeable
 -64 ErrUnavailable
 -65 ErrUnknown
 -66 ErrKilled
 -67 ErrInUse
 -68 ErrEndOfData
 -69 ErrMemKeyInvalid
 -70 ErrMemSgnInvalid
 -71 ErrMemMsgVerify
 -72 ErrMemKeyLength
 -73 ErrMemIVLength
 -74 ErrMemIVInvalid
 -75 ErrMemDecrypt
 -2920 ErrSvcSessionState

Externe Dateioperationen

Code Konstante

-20 ErrFsiNoFile
 -21 ErrFsiNoPath
 -22 ErrFsiOpenOverflow
 -23 ErrFsiAccessDenied
 -24 ErrFsiHdlInvalid
 -25 ErrFsiDriveInvalid
 -26 ErrFsiCurrentDir
 -27 ErrFsiSharingViolation
 -28 ErrFsiLockViolation
 -29 ErrFsiOpenFailed
 -31 ErrFsiReadFault
 -32 ErrFsiWriteFault
 -35 ErrFsiInvalidFormat

-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 ErrDbNoServer
- 802 ErrDbComm
- 803 ErrDbNoArea
- 804 ErrDbAreaOpen
- 805 ErrDbAreaLocked
- 806 ErrDbAreaInUse
- 807 ErrDbAreaType
- 808 ErrDbAreaPassword
- 809 ErrDbUserLimit
- 810 ErrDbServerStart
- 811 ErrDbUserInvalid
- 813 ErrDbUserSelf
- 817 ErrDbAreaRollback
- 827 ErrDbAreaStandby
- 830 ErrDbAreaLockedAdmin
- 831 ErrDbAreaLockedOperation
- 832 ErrDbAreaLockedDown
- 833 ErrDbAreaLockedStandby
- 834 ErrDbAreaLockedRollback
- 835 ErrDbAreaLockedOpen
- 836 ErrDbAreaLockedNoStandbyOpen

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 ErrSckProxyConnectFailed
- 726 ErrSckProxyRead
- 727 ErrSckProxyWrite
- 732 ErrSckProxyFailed
- 733 ErrSckProxyAuthFailed
- 741 ErrSckTlsConnect
- 745 ErrSckTlsCertificateVerify

Netzwerkinformations-Fehler

Code Konstante

- 705 ErrNetNoHost
- 706 ErrNetCreate
- 709 ErrNetSelect
- 710 ErrNetRead
- 712 ErrNetReadLess
- 713 ErrNetWrite
- 719 ErrNetIcmpType
- 720 ErrNetIcmpID

Fehler von binären Objekten

Code Konstante

- 1501 ErrBinNameInvalid
- 1502 ErrBinNoPath
- 1503 ErrBinNoFile
- 1504 ErrBinNoData
- 1505 ErrBinNoLock
- 1506 ErrBinLocked
- 1507 ErrBinExists
- 1508 ErrBinDirNotEmpty
- 1509 ErrBinData
- 1510 ErrBinOperation
- 1512 ErrBinDecryption

Fehler von Storage-Objekten

Code Konstante

- 1501 ErrStoNameInvalid
- 1502 ErrStoNoPath
- 1503 ErrStoNoFile
- 1504 ErrStoNoData
- 1506 ErrStoLocked
- 1510 ErrStoOperation
- 1511 ErrStoInvalidFormat

Fehler von Validierungsbefehlen

Code Konstante

-1501 __ErrVldNameInvalid

-1503 __ErrVldNoFile

-1505 __ErrVldNoLock

-1506 __ErrVldLocked

-1507 __ErrVldExists

TAPI-Fehler

Code Konstante

-1801 __ErrTapiUnknown

-1802 __ErrTapiVersion

-1803 __ErrTapiDevName

-1804 __ErrTapiInUse

-1805 __ErrTapiDialString

-1806 __ErrTapiDialTimeout

-1807 __ErrTapiInstall

-1808 __ErrTapiReinit

-1809 __ErrTapiMemory

-1810 __ErrTapiFailed

-1811 __ErrTapiUnavail

-1812 __ErrTapiMediaMode

-1813 __ErrTapiBusy

-1814 __ErrTapiBadAddr

-1815 __ErrTapiNoConnect

-1816 __ErrTapiReject

-1817 __ErrTapiCallState

-1819 __ErrTapiCallId

-1820 __ErrTapiNotOwner

-1821 __ErrTapiNoListen

Druckprozessor

Code Konstante

-9074 __ErrPpcAcrobat

-9034 __ErrPpcArgument

-9070 __ErrPpcDriver

-9054 __ErrPpcFileCreate

-9051 __ErrPpcFileOpen

-9052 __ErrPpcFileRead

-9053 __ErrPpcFileWrite

-9015 __ErrPpcGhostscript

-9081 __ErrPpcPrint

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

- 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.



ErrIgnore(int1, logic2)

Zu ignorierende Laufzeitfehler festlegen

Fehlerwert

	<u>ErrCallOld</u>	Fehler beim Aufruf von A- Prozeduren
int1	<u>ErrCnv</u>	Fehler bei Typkonvertierung
	<u>ErrDecimal</u>	Fehler bei Dezimalberechnung
	<u>ErrStringOverflow</u>	Zeichenkettenüberlauf
logic2	Laufzeitfehlermeldung ausschalten (<u>true</u>) oder einschalten (<u>false</u>)	

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.

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
```

Mögliche Laufzeitfehler

ErrValueInvalid In (alpha2) wurden eine unbekannte Sprachkennung angegeben.

ErrPos() : int



Fehlerposition abfragen

Resultat int Label

Siehe Verwandte
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.

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.

ErrThrowProc()



Fehler auslösende Funktion

Resultat alpha Prozedur- und Funktionsname

Siehe Verwandte Befehle, try, try

Wird ein try- oder try-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 try-Block beginnt.

Beispiel:

```
sub myFunc{ try { ... ErrSet(_ErrGeneric); ... }}main{ try { myFunc(); } if (E
```



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,

Siehe try, ErrTryCatch(),

Fehlerbehandlung

(Blog)

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.

XmlError(int1) : alpha



Weitere Informationen zu eine 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.

Allgemeine Fehlerkonstanten

Siehe Alle
Befehle

<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

_ErrAll
Symbolischer Wert für alle Fehler bei ErrTryIgnore()
Wert -2.147.483.647
/ 0x80000001
Siehe Verwandte
Befehle
Option von ErrTryIgnore() - Es werden alle Fehler ignoriert.

_ErrData
Datenfehler
Wert -5

Verwandte
Siehe Befehle,
ErrGet()

Kategorie : Allgemein

Ursache : Beim Verarbeiten der Daten ist ein allgemeiner Fehler aufgetreten.

_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.

_ErrGeneric

Ein allgemeiner Fehler ist aufgetreten

Wert -1

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Allgemein

Ursache : Ist das Resultat des Befehls DbKeyRebuild() _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.

_ErrOk
Kein Fehler aufgetreten
Wert 0

Verwandte
Siehe Befehle,
ErrGet()
Die Operation ist ohne Fehler durchgeführt worden.

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.

_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.

Konstanten für Laufzeitfehler

Siehe Alle
Befehle

<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>ErrFrameDiffers</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

ErrValueOverflow Wert zu groß oder zu klein

ErrValueRange Wert außerhalb des zulässigen Wertebereichs

_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 Änderung 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.

_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.

_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.

_ErrCodeMissing
Prozedurcode fehlt
Wert -161

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Laufzeitfehler

Ursache : Bei der Durchführung einer Prozedur ist der Code nicht vorhanden.

_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).

_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.

_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.

_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.

_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.

_ErrFileInvalid

Datei ungültig

Wert -194

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Die angegebene Datei kann bei der Operation nicht verwendet werden.

_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.

_ErrFrameDiffers
Frame abweichend
Wert -202

Verwandte
Siehe Befehle,
ErrGet(),
with

Kategorie : Laufzeitfehler

Ursache : Ein Frame, welcher durch ein with-Statement referenziert wird, wurde nach dem Übersetzen der Prozedur geändert.

_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.

_ErrIllegalOp
Unzulässige Anweisung
Wert -205

Siehe Verwandte
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.

_ErrLinkInvalid
Verknüpfung ungültig
Wert -193

Verwandte
Siehe Befehle,
ErrGet()

Kategorie : Laufzeitfehler

Ursache : Die angegebene Verknüpfung passt nicht zur Zielfile oder zum angegebenen Filter.

_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()

_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.

_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.

_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.

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.

_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.

_ErrNoKeyFld

Schlüsselfeld ist nicht in der Datenstruktur vorhanden

Wert -189

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Das in dem Schlüssel angegebene Schlüsselfeld ist nicht in der
Datenstruktur vorhanden.

_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.

_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.

_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.

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.

_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.

_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.

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.

_ErrSelValueSet

Kombination von zwei Selektionen mit Wertmenge

Wert -195

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Laufzeitfehler

Ursache : Es sollen zwei Selektionen mit _SelUnion, _SelInter oder _SelMinus miteinander kombiniert werden, wobei eine von von beiden Selektionen eine Wertmenge enthält.

_ErrServerTerm
Server wurde beendet
Wert -166

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Laufzeitfehler

Ursache : Während RmtCall() oder SelRun(..., _SelServer...) wurde der Server
beendet, die Datenbank geschlossen, oder der Benutzer <Intern>
abgemeldet wurde.

_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.

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.

_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

Ursache : Der benutzte Wert liegt außerhalb des für die Funktion gültigen Wertebereichs. Der Fehler wird auch dann erzeugt, wenn eine 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()
- DbControl()
- DbConnect()
- DbDisconnect()
- WseInfo()

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

_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

_rExists
Datensatz existiert bereits
Wert 6

Verwandte
Befehle,
Siehe RecInsert(),
RecReplace(),
SelRecInsert(),
ErrGet()

Kategorie : Datensatzoperation

Ursache : Der Datensatz konnte nicht eingefügt oder rückgespeichert werden, da
: bereits ein Satz mit einem identischen eindeutigen Schlüsselwert
existiert.

_rFailed
Zugriff auf temporären Baum nicht möglich
Wert 12

Verwandte
Befehle,
RecRead(),
Siehe RecLink(),
SelRecInsert(),
SelRecDelete(),
ErrGet()

Kategorie : Datensatzoperation

Ursachen : Es wird auf einen temporären Baum zugegriffen, der nicht mehr 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.

_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.

_rLimitReached
Datensatzlimit überschritten
Wert 11

Verwandte
Siehe Befehle,
ErrGet()

Kategorie : Datensatzoperation

Ursache : Die maximale Anzahl der zu ermittelnden Datensätze wurde überschritten.

Die Hauptergebnismenge enthält trotz des Fehlerwertes die gefundenen Datensätze bis zum Limit.

_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.

_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.

_rNoKey

Schlüssel nicht vorhanden / Kein leerer Schlüssel vorhanden

Wert 3

Verwandte

Befehle,

RecRead(),

Siehe SelRead(),

SelDelete(),

DbKeyRebuild(),

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 DbKeyRebuild() 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.

_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 :
- Bei einem Zugriff über eine Verknüpfung existiert kein verknüpfter Datensatz.
 - Bei RecReplace() wurde kein Datensatz mit der aktuellen Datensatz-ID gefunden.

Wird der Wert von einer RmtDataRead()-Anweisung zurückgegeben, konnte das Datenobjekt nicht gefunden werden.

_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.

_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.

_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.

Konstanten für Datenbankfehler

Siehe Alle
Befehle

<u>ErrDbAreaInUse</u>	Datenbank ist vorübergehend gesperrt.
<u>ErrDbAreaLocked</u>	Datenbank ist gesperrt.
<u>ErrDbAreaLockedAdmin</u>	Datenbank ist durch den Administrator gesperrt.
<u>ErrDbAreaLockedDown</u>	Datenbankprozess wird beendet.
<u>ErrDbAreaLockedNoStandbyOpen</u>	Standby-System steht nicht zur Verfügung.
<u>ErrDbAreaLockedOpen</u>	Sperre für Login gesetzt.
<u>ErrDbAreaLockedOperation</u>	Datenbank ist durch eine Serviceoperation gesperrt.
<u>ErrDbAreaLockedRollback</u>	Datenbank ist gesperrt (Sperre für Rollback).
<u>ErrDbAreaLockedStandby</u>	Datenbank im Standby-Modus.
<u>ErrDbAreaOpen</u>	Fehler beim Öffnen der Datenbank.
<u>ErrDbAreaOperationDenied</u>	Operation nicht erlaubt.
<u>ErrDbAreaPassword</u>	Serverkennwort nicht korrekt.
<u>ErrDbAreaRollback</u>	Datenbank ist im Rollback-Modus.
<u>ErrDbAreaStandby</u>	Datenbank ist im Standby-Modus.
<u>ErrDbAreaType</u>	Datenbankversion ungültig.
<u>ErrDbComm</u>	Allgemeiner Kommunikationsfehler.
<u>ErrDbNoArea</u>	Datenbank nicht vorhanden.
<u>ErrDbNoServer</u>	Server nicht vorhanden.
<u>ErrDbServerStart</u>	Fehler beim Serverstart.
<u>ErrDbUserInvalid</u>	Datenbankbenutzer ist ungültig.
<u>ErrDbUserLimit</u>	Maximale Benutzerzahl ist erreicht.
<u>ErrDbUserSelf</u>	Ausloggen des eigenen Benutzers.

_ErrDbAreaInUse
Datenbank ist vorübergehend gesperrt
Wert -806

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Es konnte keine genaue Ursache ermittelt werden.

_ErrDbAreaLocked
Datenbank ist gesperrt
Wert -805

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist in exklusiver Benutzung.

_ErrDbAreaLockedAdmin

Datenbank ist durch den Administrator gesperrt

Wert -830

Verwandte

Siehe Befehle,

DbConnect(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist durch den Administrator mit einer Login-Sperre versehen worden.

_ErrDbAreaLockedDown
Datenbankprozess wird beendet
Wert -832

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Der Datenbankprozess wird gerade beendet. Die Datenbank kann zu einem späteren Zeitpunkt wieder geöffnet werden.

_ErrDbAreaLockedNoStandbyOpen
Standby-System steht nicht zur Verfügung
Wert -836

Verwandte

Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : In den Einstellungen der Datenbank ist eine Sperre eingetragen, für den
Fall, dass das Standby-System nicht zur Verfügung steht (siehe
Einstellungen der Datenbank).

_ErrDbAreaLockedOpen

Sperre für Login gesetzt

Wert -835

Verwandte

Siehe Befehle,

DbConnect(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist gesperrt. Die Sperre kann über die Web-Administration wieder freigegeben werden.

_ErrDbAreaLockedOperation

Datenbank ist durch eine Serviceoperation gesperrt

Wert -831

Verwandte

Siehe Befehle,

DbConnect(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist durch eine Serviceoperation (zum Beispiel eine Datenbankoptimierung) gesperrt.

_ErrDbAreaLockedRollback
Datenbank ist gesperrt (Sperre für Rollback)
Wert -834

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist nicht abgeschlossen, ein Rollback kann aufgrund einer Rollback-Sperre nicht durchgeführt werden. Die Rollback-Sperre kann über die Web-Administration aufgehoben werden.

_ErrDbAreaLockedStandby
Datenbank im Standby-Modus
Wert -833

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank kann nicht geöffnet werden, da der andere
Server des Hot-Standby-Systems die Datenbank im aktiven Betrieb hat.

_ErrDbAreaOpen
Fehler beim Öffnen der Datenbank
Wert -804

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die Datenbank konnte auf dem Zielsystem nicht geöffnet werden.

_ErrDbAreaOperationDenied

Operation nicht erlaubt

Wert -821

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Datenbankoperation

Ursache : Die Operation ist nicht erlaubt.

_ErrDbAreaPassword
Serverkennwort nicht korrekt
Wert -808

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist per Serverkennwort geschützt, wobei
dieses nicht korrekt angegeben wurde.

_ErrDbAreaRollback
Datenbank ist im Rollback-Modus
Wert -817

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist im Rollback-Modus und kann nicht geöffnet werden.

_ErrDbAreaStandby
Datenbank ist im Standby-Modus
Wert -827

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist im Standby-Modus und kann nicht geöffnet werden.

_ErrDbcAreaType
Datenbankversion ungültig
Wert -807

Verwandte
Siehe Befehle,
DbcConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die Version der zu öffnenden Datenbank ist nicht korrekt.

_ErrDbComm
Allgemeiner Kommunikationsfehler
Wert -802

Verwandte Befehle,
Siehe ErrGet(), DbConnect(),
_DbConnectOpErrorCode
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 _DbConnectOpErrorCode gesetzt ist. Ist die Option nicht gesetzt, kommt es im Standard- und Advanced-Client beim Zugriff auf die verbundene Datenbank zu einem Verbindungsabbruch.

_ErrDbNoArea
Datenbank nicht vorhanden
Wert -803

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Die zu öffnende Datenbank ist auf dem Zielsystem nicht eingetragen oder
nicht vorhanden.

Kontakt

_ErrDbNoServer
Server nicht vorhanden
Wert -801

Verwandte
Siehe Befehle,
DbConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Es konnte keine Verbindung mit dem Datenbankserver hergestellt werden. Möglicherweise verhindert eine Firewall den Kontakt. Zur Kommunikation muss auf dem Server der Port 4722 offen sein.

_ErrDbaserverRelease

Die Server-Version kleiner als die Client-Version

Wert -823

Verwandte

Siehe Befehle,

DbaserverConnect(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Der Fehler wird zurückgegeben, wenn die Server-Version kleiner als die Version des zu verbindenden Clients ist.

_ErrDbaserverStart
Fehler bei Serverstart
Wert -810

Verwandte
Siehe Befehle,
DbaserverConnect(),
ErrGet()

Kategorie : Datenbankoperation

Ursache : Der CONZEPT 16-Server konnte den für die Datenbank eingetragenen
Puffer nicht anlegen.

_ErrDbidUserInvalid
Datenbankbenutzer ist ungültig
Wert -811

Verwandte

Befehle,

Siehe DbConnect(),

UserClear(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Der angegebene Benutzer ist nicht vorhanden, das Kennwort ist nicht
: korrekt oder über den Benutzer ist kein Datenbankzugriff von einer
anderen Datenbank erlaubt.

Kontakt

ErrDbidUserLimit

Maximale Benutzeranzahl des CONZEPT 16-Servers ist erreicht

Wert -809

Verwandte

Siehe Befehle,

DbConnect(),

ErrGet()

Kategorie : Datenbankoperation

Ursache : Das Benutzerlimit des Servers ist erreicht oder die
Cachegröße der Zieldatenbank ist für die Benutzermenge
unzureichend.

_ErrDbidUserSelf
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.

Konstanten für Verarbeitungsfehler

Siehe Alle
Befehle

<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

_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.

_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.

_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.

_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.

_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

_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.

_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.

_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

_ErrLocked

Objekt ist gesperrt

Wert -59

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Das Objekt ist von einem anderen Benutzer gesperrt.

_ErrMemIVInvalid
Initialisierungsvektor ungültig
Wert -74

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Der bei MemEncrypt() oder MemDecrypt() angegebene Initialisierungsvektor ist ungültig. Dies ist beispielsweise der Fall, wenn sich der Initialisierungsvektor nicht aus der hexadezimal bzw. Base64-kodierten Zeichenkette dekodieren lässt.

ErrMemIVLength
Initialisierungsvektor zu kurz oder zu lang
Wert -73

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Der bei MemEncrypt() oder MemDecrypt() angegebene
Initialisierungsvektor ist zu kurz oder zu lang für den ausgewählten
Verschlüsselungsalgorithmus.

_ErrMemKeyInvalid

Schlüssel ungültig

Wert -69

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Der bei MemSign(), MemVerify(), MemEncrypt() oder MemDecrypt() angegebene Schlüssel ist ungültig. Dies ist beispielsweise der Fall, wenn der Schlüssel im falschen Format angegeben ist oder sich nicht aus der hexadezimal bzw. Base64-kodierten Zeichenkette dekodieren lässt.

ErrMemKeyLength
Schlüssel zu kurz oder zu lang
Wert -72

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Der bei MemSign(), MemEncrypt() oder MemDecrypt() angegebene Schlüssel ist zu kurz für die zu signierende Nachricht oder nicht die passende Länge für den ausgewählten Verschlüsselungsalgorithmus hat.

_ErrMemMsgVerify
Signatur passt nicht zur Nachricht
Wert -71

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Die bei MemVerify() angegebene Nachricht konnte nicht verifiziert werden, da die Signatur nicht zur Nachricht und dem öffentlichen Schlüssel passt.

_ErrMemSgnInvalid

Signatur ungültig

Wert -70

Verwandte

Siehe Befehle,

ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Die bei MemVerify() angegebene Signatur ist ungültig.

_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.

_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.

ErrRange
Außerhalb des Bereichs
Wert -51

Verwandte
Siehe Befehle,
ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Bei PrtJobClose() wurde keine zu druckende Seite ausgewählt, oder die
Startseite ist größer als die Endseite.

ErrSvcSessionState
Zustand der Session nicht korrekt
Wert -2920

Verwandte

Siehe Befehle,
ErrGet()

Kategorie : Verarbeitungsfehler

Ursache : Eine Session kann nur dann angelegt bzw. gelöscht werden, wenn die
ausgeführte Prozedur noch keine Session bzw. eine Session besitzt.
Ebenso kann nur dann eine Session der ausführenden Prozedur
zugewiesen werden, wenn diese Prozedur noch keine Session besitzt.

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.

_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

Ursache : Die Funktion kann nicht ausgeführt werden, da das Objekt oder andere Ressourcen, die benötigt werden, nicht zur Verfügung stehen.