

## Kontakt

### Eigenschaften von System-Objekten

In dieser Liste sind alle Eigenschaften von System-Objekten aufgeführt. [Liste sortiert](#)

[nach Gruppen](#),

Siehe [Alphabetische](#)  
[Liste aller](#)  
[Eigenschaften](#)

- [AttribCount](#)
- [BaseAddress](#)
- [CertificateError](#)
- [Charset](#)
- [CharsetOS](#)
- [ChartArea](#)
- [ChartBorderWidth](#)
- [ChartColBkg](#)
- [ChartColBorder](#)
- [ChartHeight](#)
- [ChartLegendColBkg](#)
- [ChartLegendColBorder](#)
- [ChartLegendKeyGap](#)
- [ChartLegendKeySize](#)
- [ChartLegendPos](#)
- [ChartNumFmtThousandSep](#)
- [ChartPieAngle](#)
- [ChartPieColEdge](#)
- [ChartPieColJoin](#)
- [ChartPieDepth](#)
- [ChartPieInnerRadius](#)
- [ChartPieLabelFont](#)
- [ChartPieShading](#)
- [ChartPieShadowMode](#)
- [ChartPieStartAngle](#)
- [ChartPyramidElevation](#)
- [ChartPyramidLayerGap](#)
- [ChartPyramidRotation](#)
- [ChartPyramidStyleData](#)
- [ChartPyramidTubeHeight](#)
- [ChartPyramidTubeRadius](#)
- [ChartPyramidTwist](#)
- [ChartSurfaceAmbient](#)
- [ChartSurfaceColContour](#)
- [ChartSurfaceColSurfaceAxis](#)
- [ChartSurfaceColSurfaceData](#)
- [ChartSurfaceContinuous](#)
- [ChartSurfaceDiffuse](#)
- [ChartSurfaceElevation](#)
- [ChartSurfaceLegendLength](#)
- [ChartSurfacePerspective](#)
- [ChartSurfacePlotHeight](#)
- [ChartSurfaceRotation](#)

- [ChartSurfaceSamplesX](#)
- [ChartSurfaceSamplesY](#)
- [ChartSurfaceShadingMode](#)
- [ChartSurfaceShine](#)
- [ChartSurfaceSmooth](#)
- [ChartSurfaceSpecular](#)
- [ChartSurfaceTitleFontX](#)
- [ChartSurfaceTitleFontY](#)
- [ChartSurfaceTitleFontZ](#)
- [ChartSurfaceTitleX](#)
- [ChartSurfaceTitleY](#)
- [ChartSurfaceTitleZ](#)
- [ChartSurfaceTwist](#)
- [ChartSurfaceWireWidth](#)
- [ChartTitleArea](#)
- [ChartTitleColBkg](#)
- [ChartTitleColFg](#)
- [ChartTitleFont](#)
- [ChartTitleText](#)
- [ChartWidth](#)
- [ChartXYAxisColY](#)
- [ChartXYAxisOffsetY](#)
- [ChartXYAxisTitleAlignY](#)
- [ChartXYAxisTitleY](#)
- [ChartXYAxisY](#)
- [ChartXYBarGap](#)
- [ChartXYBarShading](#)
- [ChartXYBarShape](#)
- [ChartXYColBkg](#)
- [ChartXYColBkgAlt](#)
- [ChartXYColBorder](#)
- [ChartXYColData](#)
- [ChartXYColGridX](#)
- [ChartXYColGridY](#)
- [ChartXYColTrend](#)
- [ChartXYDepth](#)
- [ChartXYDepthGap](#)
- [ChartXYGapDash](#)
- [ChartXYLabelAngleX](#)
- [ChartXYLabelColData](#)
- [ChartXYLabelColSum](#)
- [ChartXYLabelFontData](#)
- [ChartXYLabelFontX](#)
- [ChartXYLabelFontY](#)
- [ChartXYLabelRotData](#)
- [ChartXYLegendText](#)
- [ChartXYLineSymbol](#)
- [ChartXYLineSymbolParam](#)
- [ChartXYLineSymbolSize](#)
- [ChartXYLineWidth](#)
- [ChartXYMinTickIncY](#)

- [ChartXYStyleData](#)
- [ChartXYStyleLabel](#)
- [ChartXYSwapXY](#)
- [ChartXYTitleAlignY](#)
- [ChartXYTitleFontX](#)
- [ChartXYTitleFontY](#)
- [ChartXYTitleX](#)
- [ChartXYTitleY](#)
- [ChartXYTrendDegree](#)
- [ChartXYTrendType](#)
- [ChildCount](#)
- [CodepageOS](#)
- [Compression](#)
- [ContentLength](#)
- [CpnSetupFlagsCln](#)
- [CpnSetupFlagsDoc](#)
- [CpnSetupFlagsSrv](#)
- [Created](#)
- [CreatedUser](#)
- [CteNodeSepAttrib](#)
- [CteNodeSepPath](#)
- [Custom](#)
- [DisplayRaisingDelay](#)
- [ErrCode](#)
- [ErrLine](#)
- [ErrPos](#)
- [ErrProc](#)
- [ErrSource](#)
- [ErrSourceLine](#)
- [ErrText](#)
- [FileFilter](#)
- [FileFilterNum](#)
- [FileNameExt](#)
- [Flags](#)
- [FsiError](#)
- [FullName](#)
- [Hash](#)
- [HdlCount](#)
- [HostName](#)
- [HttpHeader](#)
- [HttpParameters](#)
- [ID](#)
- [JobData](#)
- [JobErrorCode](#)
- [JobID](#)
- [JobMsxReadQ](#)
- [JobMsxWriteQ](#)
- [JobProcesses](#)
- [JobProcExtended](#)
- [JobSckHandle](#)
- [JobStatus](#)

- [JobThreads](#)
- [LclCurrDecimal](#)
- [LclCurrFract](#)
- [LclCurrFractIntl](#)
- [LclCurrGroup](#)
- [LclCurrNegSignPos](#)
- [LclCurrNegSpaceSep](#)
- [LclCurrNegSymPrec](#)
- [LclCurrPosSignPos](#)
- [LclCurrPosSpaceSep](#)
- [LclCurrPosSymPrec](#)
- [LclCurrSymbol](#)
- [LclCurrSymbolIntl](#)
- [LclCurrTSep](#)
- [LclDateDayLZero](#)
- [LclDateDayN](#)
- [LclDateDayNS](#)
- [LclDateLFormat](#)
- [LclDateLOrder](#)
- [LclDateMonthLZero](#)
- [LclDateMonthN](#)
- [LclDateMonthNS](#)
- [LclDateSCentury](#)
- [LclDateSep](#)
- [LclDateSFormat](#)
- [LclDateSOrder](#)
- [LclNumDecimal](#)
- [LclNumFract](#)
- [LclNumGroup](#)
- [LclNumLZero](#)
- [LclNumNegMode](#)
- [LclNumNegSign](#)
- [LclNumPosSign](#)
- [LclNumTSep](#)
- [LclTimeHourMode](#)
- [LclTimeLFormat](#)
- [LclTimeLZero](#)
- [LclTimeMarkPos](#)
- [LclTimeSep](#)
- [LclTimeSepAM](#)
- [LclTimeSepPM](#)
- [LclTimeSFormat](#)
- [Len](#)
- [LogicalProcessors](#)
- [Method](#)
- [Modified](#)
- [ModifiedUser](#)
- [Name](#)
- [ObjectsGDI](#)
- [ObjectsGDILimit](#)
- [ObjectsUser](#)

- [ObjectsUserLimit](#)
- [OdbcClmBufferLen](#)
- [OdbcClmCatalog](#)
- [OdbcClmDataDefault](#)
- [OdbcClmDataType](#)
- [OdbcClmDataTypeName](#)
- [OdbcClmDecimalDigits](#)
- [OdbcClmName](#)
- [OdbcClmNullable](#)
- [OdbcClmNumPrecRadix](#)
- [OdbcClmOrdinal](#)
- [OdbcClmRemarks](#)
- [OdbcClmSchema](#)
- [OdbcClmSize](#)
- [OdbcClmTable](#)
- [OdbcConCfcOdbcApi](#)
- [OdbcConCfcOdbcSql](#)
- [OdbcConCfcSagCli](#)
- [OdbcConDatabaseName](#)
- [OdbcConDbmsName](#)
- [OdbcConDbmsVersion](#)
- [OdbcConDriverName](#)
- [OdbcConDriverOdbcVersion](#)
- [OdbcConDriverVersion](#)
- [OdbcConDSN](#)
- [OdbcConOdbcVersion](#)
- [OdbcConReadOnly](#)
- [OdbcConServerName](#)
- [OdbcConUserName](#)
- [OdbcErrSqlMessage](#)
- [OdbcErrSqlNativeCode](#)
- [OdbcErrSqlResult](#)
- [OdbcErrSqlState](#)
- [OdbcResCountClm](#)
- [OdbcResCountRow](#)
- [OdbcTblCatalog](#)
- [OdbcTblName](#)
- [OdbcTblRemarks](#)
- [OdbcTblSchema](#)
- [OdbcTblType](#)
- [Parent](#)
- [PathAllAppData](#)
- [PathAllAutoStart](#)
- [PathAllDesktop](#)
- [PathAllPrograms](#)
- [PathAppData](#)
- [PathAppProg](#)
- [PathAppRoot](#)
- [PathAutoStart](#)
- [PathClient](#)
- [PathDesktop](#)

- PathFonts
- PathMyDocuments
- PathMyMusic
- PathMyPictures
- PathMyVideo
- PathOffcAccess
- PathOffcExcel
- PathOffcOutlook
- PathOffcPowerPoint
- PathOffcWord
- PathPfmData
- PathPfmRoot
- PathProgramFiles
- PathProgramFilesCommon
- PathPrograms
- PathSystem
- PathTemp
- PathWindows
- PdfAttachCount
- PdfAuthor
- PdfCompany
- PdfCreator
- PdfKeywords
- PdfPageCount
- PdfPageHeight
- PdfPageRotation
- PdfPageWidth
- PdfPrintOrder
- PdfProducer
- PdfResolution
- PdfSubject
- PdfTitle
- PdfUserRights
- PdfVersion
- PhysicalMemoryMB
- Platform
- PriorityIdle
- ProcCacheKB
- ProcCacheLimitKB
- ProcCurrent
- ProcCurrentFull
- ProcCurrentSub
- ProcessArchitecture
- ProcessExitCode
- ProcessID
- ProcessMemory
- ProcessMemoryKB
- ProcessMemoryLimitMB
- Protocol
- ProxyAuthorization
- ProxyAuthType

- Size
- SizeDba
- SizeDba64
- SizeOrg
- SizeOrg64
- StatusCode
- StopRequest
- StorageID
- SvcCallDelay
- SvcCallTime
- SvcDescription
- SvcName
- SvcSckHandle
- SvcSessionID
- TapiFlags
- TapiModuleFilename
- TapiNameComputer
- TapiOwnerRequest
- TapiUserName
- TerminalSession
- TerminalSessionID
- TimeExternal
- Type
- TypeMime
- TypeUser
- URI
- Validated
- ValidatedUser
- ValueAlpha
- ValueBigInt
- ValueCaltime
- ValueColor
- ValueDate
- ValueDecimal
- ValueEvent
- ValueFloat
- ValueFont
- ValueInt
- ValueLogic
- ValuePoint
- ValueRange
- ValueRect
- ValueRtfTab
- ValueTime
- Version

## Kontakt

### Eigenschaften von System-Objekten - A bis E

In dieser Liste sind alle Eigenschaften von A bis E von System-Objekten aufgeführt. Liste sortiert

nach Gruppen,

Siehe Alphabetische

Liste aller  
Eigenschaften

- [AttribCount](#)
- [BaseAddress](#)
- [CertificateError](#)
- [Charset](#)
- [CharsetOS](#)
- [ChartArea](#)
- [ChartBorderWidth](#)
- [ChartColBkg](#)
- [ChartColBorder](#)
- [ChartHeight](#)
- [ChartLegendColBkg](#)
- [ChartLegendColBorder](#)
- [ChartLegendKeyGap](#)
- [ChartLegendKeySize](#)
- [ChartLegendPos](#)
- [ChartNumFmtThousandSep](#)
- [ChartPieAngle](#)
- [ChartPieColEdge](#)
- [ChartPieColJoin](#)
- [ChartPieDepth](#)
- [ChartPieInnerRadius](#)
- [ChartPieLabelFont](#)
- [ChartPieShading](#)
- [ChartPieShadowMode](#)
- [ChartPieStartAngle](#)
- [ChartPyramidElevation](#)
- [ChartPyramidLayerGap](#)
- [ChartPyramidRotation](#)
- [ChartPyramidStyleData](#)
- [ChartPyramidTubeHeight](#)
- [ChartPyramidTubeRadius](#)
- [ChartPyramidTwist](#)
- [ChartSurfaceAmbient](#)
- [ChartSurfaceColContour](#)
- [ChartSurfaceColSurfaceAxis](#)
- [ChartSurfaceColSurfaceData](#)
- [ChartSurfaceContinuous](#)
- [ChartSurfaceDiffuse](#)
- [ChartSurfaceElevation](#)
- [ChartSurfaceLegendLength](#)
- [ChartSurfacePerspective](#)
- [ChartSurfacePlotHeight](#)
- [ChartSurfaceRotation](#)

- [ChartSurfaceSamplesX](#)
- [ChartSurfaceSamplesY](#)
- [ChartSurfaceShadingMode](#)
- [ChartSurfaceShine](#)
- [ChartSurfaceSmooth](#)
- [ChartSurfaceSpecular](#)
- [ChartSurfaceTitleFontX](#)
- [ChartSurfaceTitleFontY](#)
- [ChartSurfaceTitleFontZ](#)
- [ChartSurfaceTitleX](#)
- [ChartSurfaceTitleY](#)
- [ChartSurfaceTitleZ](#)
- [ChartSurfaceTwist](#)
- [ChartSurfaceWireWidth](#)
- [ChartTitleArea](#)
- [ChartTitleColBkg](#)
- [ChartTitleColFg](#)
- [ChartTitleFont](#)
- [ChartTitleText](#)
- [ChartWidth](#)
- [ChartXYAxisColY](#)
- [ChartXYAxisOffsetY](#)
- [ChartXYAxisTitleAlignY](#)
- [ChartXYAxisTitleY](#)
- [ChartXYAxisY](#)
- [ChartXYBarGap](#)
- [ChartXYBarShading](#)
- [ChartXYBarShape](#)
- [ChartXYColBkg](#)
- [ChartXYColBkgAlt](#)
- [ChartXYColBorder](#)
- [ChartXYColData](#)
- [ChartXYColGridX](#)
- [ChartXYColGridY](#)
- [ChartXYColTrend](#)
- [ChartXYDepth](#)
- [ChartXYDepthGap](#)
- [ChartXYGapDash](#)
- [ChartXYLabelAngleX](#)
- [ChartXYLabelColData](#)
- [ChartXYLabelColSum](#)
- [ChartXYLabelFontData](#)
- [ChartXYLabelFontX](#)
- [ChartXYLabelFontY](#)
- [ChartXYLabelRotData](#)
- [ChartXYLegendText](#)
- [ChartXYLineSymbol](#)
- [ChartXYLineSymbolParam](#)
- [ChartXYLineSymbolSize](#)
- [ChartXYLineWidth](#)
- [ChartXYMinTickIncY](#)

- [ChartXYStyleData](#)
- [ChartXYStyleLabel](#)
- [ChartXYSwapXY](#)
- [ChartXYTitleAlignY](#)
- [ChartXYTitleFontX](#)
- [ChartXYTitleFontY](#)
- [ChartXYTitleX](#)
- [ChartXYTitleY](#)
- [ChartXYTrendDegree](#)
- [ChartXYTrendType](#)
- [ChildCount](#)
- [CodepageOS](#)
- [Compression](#)
- [ContentLength](#)
- [CpnSetupFlagsCln](#)
- [CpnSetupFlagsDoc](#)
- [CpnSetupFlagsSrv](#)
- [Created](#)
- [CreatedUser](#)
- [CteNodeSepAttrib](#)
- [CteNodeSepPath](#)
- [Custom](#)
- [DisplayRaisingDelay](#)
- [ErrCode](#)
- [ErrLine](#)
- [ErrPos](#)
- [ErrProc](#)
- [ErrSource](#)
- [ErrSourceLine](#)
- [ErrText](#)

## Kontakt

**AttribCount**  $\sqsubset$  Anzahl der Attribute bei CteNode-  
Objekten Typ int

Siehe Liste, Objekte, ChildCount

Mit dieser Eigenschaft kann die Anzahl der Attribute eines CteNode-Objekts ermittelt werden.

```
// Anzahl der Attributemente ermittelntCount # tNode->spAttribCount;
```

## Kontakt

---

**BaseAddress**

**Adresse des Speicherbereichs**

**Typ bigint**

**Siehe Liste, SysPropGet()**

Dieser Wert ist die Adresse des Speicherbereichs des Memory-Objekts im aktuellen Prozess. Bei 32-Bit-Systemen werden nur die unteren 32 Bit des Wertes verwendet. Der Wert kann beispielsweise an eine DLL übergeben werden, um einen direkten Zugriff auf den Speicherinhalt durchzuführen.

**Die Eigenschaft kann nur gelesen werden.**

## Kontakt

**CertificateError**  Fehlerwert bei der  
Zertifikatsüberprüfung Typ int

Siehe Liste, Objekte

In dieser Eigenschaft kann der Fehlerwert, der bei einer fehlgeschlagenen Zertifikatsüberprüfung aufgetreten ist, abgefragt werden. Der Fehlerwert kann mit der Anweisung ErrMapText() in eine Fehlermeldung gewandelt werden.

Beispiel:

```
tErrCode # _Sys->spCertificateError;tErrMsg # ErrMapText(tErrCode, 'EN', _ErrMapX509);
```

### Charset

#### Verwendeter Zeichensatz

Typ int

Siehe Liste, Objekte

Diese Eigenschaft steht bei den Memory-Objekten, PrintJob-Objekten und dem System-Objekt zur Verfügung.

Beim System-Objekt kann über die Eigenschaft die von der Applikation verwendete Zeichensatz ermittelt werden. Das Resultat kann mit den unten stehenden Konstanten verglichen werden. In diesem Objekt kann die Eigenschaft nicht gesetzt werden.

Enthält ein Memory-Objekt alphanumerische Daten, bestimmt diese Eigenschaft den zugehörigen Zeichensatz. Dieser Wert wird bei den Funktionen MemFindStr(), MemReadStr(), MemWriteStr() und MemCnv() zur Umwandlung der Zeichen verwendet.

Beim PrintJob-Objekt bestimmt die Eigenschaft den Zeichensatz, in dem die XML-Datei erzeugt wird, wenn beim PrtJobClose() PrtJobXml angegeben wird.

Folgende Werte sind definiert (siehe auch Zeichensätze):

- \_CharsetC16\_...

Dies sind die 8-Bit-Zeichensätze, in denen alphanumerische Daten innerhalb von CONZEPT 16 gespeichert werden. Es gibt drei Varianten für die Codepages 1250 (osteuropäisch), 1252 (westeuropäisch) und 1254 (türkisch). Die verwendete Variante wird anhand der aktiven Codepage oder der eingestellten Sprache in CONZEPT 16 ausgewählt.

- \_CharsetC16\_1250

Dies ist der interne Zeichensatz in der Variante für osteuropäische Sprachen. Alle Zeichen der Windows Codepage 1250 sind hier definiert.

- \_CharsetC16\_1252

Dies ist der interne Zeichensatz in der Variante für westeuropäische Sprachen. Alle Zeichen der Windows Codepage 1252 sind hier definiert.

- \_CharsetC16\_1254

Dies ist der interne Zeichensatz in der Variante für die türkische Sprache. Alle Zeichen der Windows Codepage 1254 sind hier definiert.

- \_CharsetWCP\_1250

Dies ist der Windows-Zeichensatz für osteuropäische Sprachen (Codepage 1250).

- \_CharsetWCP\_1252

Dies ist der Windows-Zeichensatz für westeuropäische Sprachen (Codepage 1252).

- \_CharsetWCP\_1254

Dies ist der Windows-Zeichensatz für die türkische Sprachen (Codepage 1254).

## Kontakt

- **\_CharsetUndefined**

**Die aktuelle Codepage wird nicht unterstützt oder es wurde der Zeichensatz eines Linux-Betriebssystems abgefragt.**

- **\_CharsetOEM**

**Dies ist der ursprüngliche Zeichensatz des IBM PCs (IBM 437), der auch unter DOS verwendet wurde.**

- **\_CharsetOEM\_852**

**Dies ist der DOS-Zeichensatz für zentraleuropäische Sprachen (Codepage 852). Er enthält alle darstellbaren Zeichen aus ISO 8859-2.**

- **\_CharsetISO\_8859\_1**

**Dies ist der 8-Bit ISO-Zeichensatz für westeuropäische Sprachen (Latin-1).**

- **\_CharsetISO\_8859\_2**

**Dies ist der 8-Bit ISO-Zeichensatz für osteuropäische Sprachen (Latin-2).**

- **\_CharsetISO\_8859\_9**

**Dies ist der 8-Bit ISO-Zeichensatz für die türkische Sprache (Latin-5).**

- **\_CharsetISO\_8859\_15**

**Dies ist ein erweiterter 8-Bit ISO-Zeichensatz für westeuropäische Sprachen (Latin-9), der unter anderem auch das Euro-Zeichen umfasst.**

- **\_CharsetHTML**

**Dieser Zeichensatz basiert auf Unicode 5.1, wobei alle Zeichen ab Codeposition 128 sowie die Zeichen &,<,>," in der Form &#...; im Text gespeichert sind.**

- **\_CharsetUTF8**

**Dieser Zeichensatz basiert auf Unicode 5.1, wobei die einzelnen Zeichen im UTF-8-Format gespeichert sind. Ein einzelnes Zeichen kann dabei 1 bis 4 Bytes belegen.**

- **\_CharsetUTF16**

**Dieser Zeichensatz basiert auf Unicode 5.1, wobei die einzelnen Zeichen im UTF-16-Format gespeichert sind. Ein einzelnes Zeichen kann dabei 2 oder 4 Bytes belegen. Dieser Wert kann zusätzlich mit zwei weiteren Konstanten kombiniert werden:**

- ♦ **\_UTFxBE**

**Der UTF-16-Text ist im Big-Endian-Format abgelegt. Standardmäßig wird der Text im Little-Endian-Format verarbeitet.**

- ♦ **\_UTFxLE**

**Der UTF-16-Text ist im Little-Endian-Format abgelegt. Dies ist der Standardwert für UTF-16.**

- ♦ **\_UTFxBOM**

## Kontakt

Beim Umwandeln in einen UTF-16-Text wird eine Byte-Order-Mark vorangestellt, der das verwendete Endian-Format enthält. Die Konstante ist nur in Kombination mit \_UTFxBE und \_UTFxLE sinnvoll.

- **\_CharsetURI**

Dieser Zeichensatz basiert auf Unicode in UTF-8-Kodierung, wobei alle Bytes außer A-Z, a-z, 0-9, Punkt, Bindestrich, Unterstrich und Tilde in hexadezimaler Notation mit vorangestelltem Prozentzeichen dargestellt werden.

## Kontakt

CharsetOS



Zeichensatz des Betriebssystems

Typ int

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Zeichensatz des Windows-Betriebssystems ermittelt werden. Der Rückgabewert kann mit den CharsetWCP...-Konstanten verglichen werden. Wird die aktuelle Codepage des Betriebssystems nicht unterstützt oder der Zeichensatz eines Linux-Betriebssystems abgefragt, gibt die Eigenschaft CharsetUndefined zurück.

## Kontakt

### ChartArea



Bereich der Grafik innerhalb des Chart-Objekts Typ  
rect

Siehe Liste, Chart

Diese Eigenschaft definiert den Bereich der Grafik innerhalb des bei ChartOpen() angegebenen Ausgabebereichs. Außerhalb der Grafik kann so zum Beispiel Platz für die Legende geschaffen werden (siehe ChartLegendPos).

In dem angegebenen Bereich wird nur die eigentliche Grafik gezeichnet. Alle Bezeichnungen, Achsen-Beschriftungen und Schatten befinden sich nicht zwingend innerhalb des Bereichs.

## Kontakt

### ChartBorderWidth

#### Rahmen des Chart-Objekts

Typ int  
Siehe Liste, Chart,

Die Eigenschaft definiert die Breite des Rahmens, der um das Chart-Objekt gezeichnet werden soll. Der Rahmen wird innerhalb des Ausgabebereichs gezeichnet. Wird ein positiver Wert angegeben, wird ein erhabener Rahmen, bei einem negativen Wert, ein eingedrückter Rahmen gezeichnet.

Soll kein Rahmen gezeichnet werden, muss 0 angegeben werden.

Beispiele:

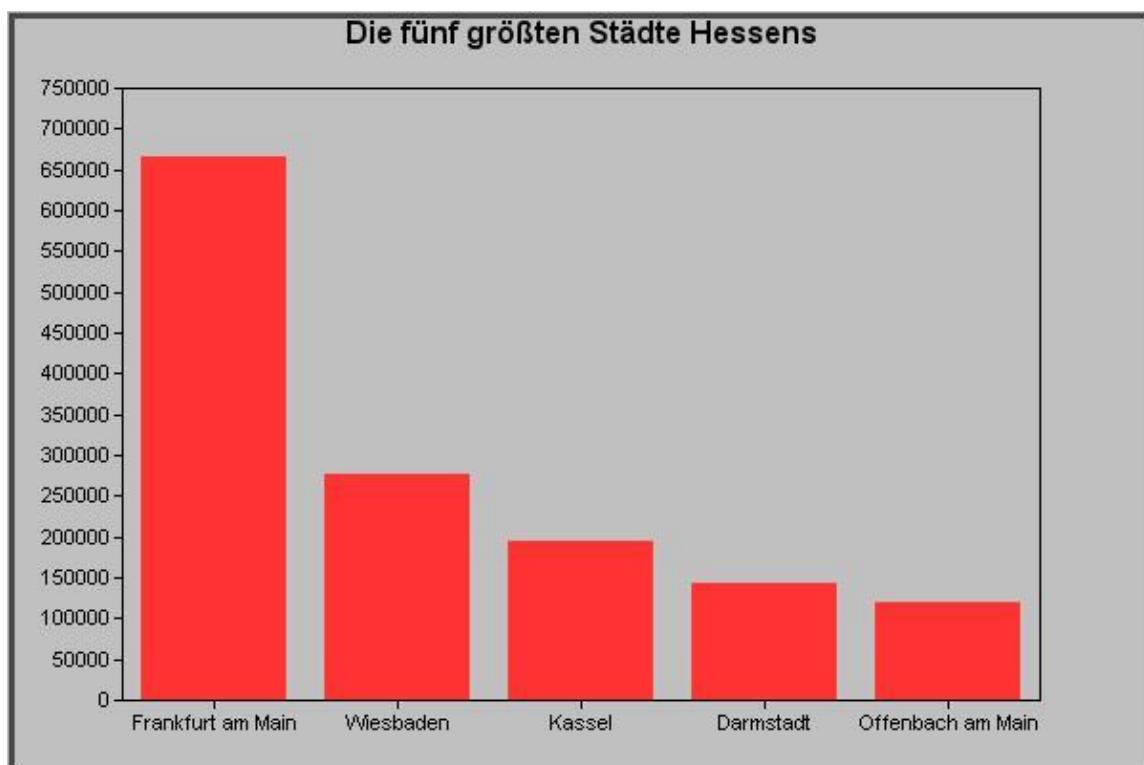
Die folgenden Grafiken wurden mit einem drei Pixel starken Rand versehen. Zur besseren Unterscheidung zum Hintergrund wurde als Hintergrundfarbe der Grafik grau gewählt.

#### Grafik mit positivem Rahmen



#### Grafik mit negativem Rahmen

## Kontakt



## Kontakt

---

**ChartColBkg**

Hintergrundfarbe des Chart-Objekts

Typ color

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Hintergrundfarbe des Chart-Objekts gesetzt und abgefragt werden.

### ChartColBorder

Randfarbe des Chart-Objekts

Typ color

Liste, Chart,

Siehe ChartBorderWidth

Mit dieser Eigenschaft kann die Farbe des Randes des Chart-Objekts gesetzt und abgefragt werden. Es wird kein Rand gezeichnet, wenn die Farbe auf ColorMake( WinTransparent, 0) oder die Eigenschaft ChartBorderWidth auf 0 gesetzt wird.

## Kontakt

**ChartHeight**



Höhe des Ausgabebereiches des Chart-Objekts

Typ int

Siehe Liste, Chart, ChartWidth, ChartOpen()

Mit dieser Eigenschaft kann die Höhe des Ausgabebereiches des Chart-Objekts gesetzt oder ermittelt werden.

## Kontakt

ChartLegendColBkg



Hintergrundfarbe der Legende Typ

color

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Hintergrundfarbe der Legende gesetzt oder abgefragt werden.

Standardmäßig ist die Hintergrundfarbe transparent.

## Kontakt

ChartLegendColBorder

Rahmenfarbe der Legende Typ

color

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Rahmenfarbe der Legende gesetzt und abgefragt werden.

Standardmäßig ist die Farbe auf schwarz gesetzt.

## Kontakt

**ChartLegendKeyGap**



**Abstand zwischen Quadrat und Text in der Legende**

**Typ** int

**Siehe** Liste, Chart

**Mit dieser Eigenschaft kann der Abstand zwischen den farbigen Quadraten und dem Text in der Legende gesetzt oder ermittelt werden. Die Angabe erfolgt in Pixel. Standardmäßig ist der Abstand auf 8 Pixel gesetzt.**

## Kontakt

**ChartLegendKeySize**



Größe der farbigen Quadrate in der Legende Typ

point

Siehe [Liste](#), [Chart](#)

Diese Eigenschaft bestimmt die Größe der farbigen Quadrate, die in der Legende gezeichnet werden. Standardmäßig werden Quadrate mit 10 x 10 Pixel gezeichnet. Es wird immer ein Quadrat gezeichnet, auch wenn eine nicht-quadratische Größe (zum Beispiel PointMake(20, 10)) angegeben wurde. Das Quadrat wird dann so groß wie möglich (in dem Beispiel 10 x 10 Pixel) in der Mitte des Bereichs gezeichnet.

**Beispiel:**

```
tHd1Chart->spChartLegendKeySize # PointMake(20, 20);
```

## Kontakt

**ChartLegendPos**

Position der Legende

Typ point

Siehe Liste, Chart

Diese Eigenschaft setzt oder ermittelt die Position der Legende im Chart-Objekt, sofern das Objekt mit einer Legende erstellt wurde (siehe ChartOpen()).

Standardmäßig wird die Legende an die Position (0, 0) im Ausgabebereich gezeichnet.

## Kontakt

ChartNumFmtThousandSep

Tausendertrennzeichen

Typ [alpha\(1\)](#)

Siehe [Liste](#), [Chart](#)

Diese Eigenschaft definiert das Tausendertrennzeichen für die Darstellung von Zahlen im [Chart](#)-Objekt. Das Zeichen '~' (Tilde) ist reserviert für kein Tausendertrennzeichen.

## Kontakt

ChartPieAngle



Winkel des Schattens / Blickwinkel auf das Torten-Diagramm Typ int

Siehe Liste, Chart

In dieser Eigenschaft kann der Winkel des Schattens (ChartPieShadowMode = true) bzw. der Blickwinkel auf das Torten-Diagramm (ChartPieShadowMode = false) gesetzt und ermittelt werden.

Bei der Darstellung mit Schatten können Winkel bis 360 Grad, bei der Darstellung als Tortendiagramm bis 90 Grad angegeben werden.

## Kontakt

**ChartPieColEdge**

Rahmenfarbe der Sektoren

Typ color

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Rahmenfarbe der Sektoren in einem Tortendiagramm gesetzt und abgefragt werden.

Standardmäßig wird kein Rahmen gezeichnet.

## Kontakt

**ChartPieColJoin**



**Farbe der Linien zwischen Sektor und Bezeichner Typ**

**color**

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft wird die Farbe der Verbindung zwischen dem Sektor und dem Bezeichner in einem Torten-Diagramm gesetzt und ermittelt.

Standardmäßig wird kein Verbinder gezeichnet.

## Kontakt

**ChartPieDepth**



Tiefe des Schatten / Höhe des Torten-Diagramms Typ int

Siehe Liste, Chart, ChartPieShadowMode

In dieser Eigenschaft kann die Tiefe des Schattens (ChartPieShadowMode = true) bzw. die Höhe des Torten-Diagramms (ChartPieShadowMode = false) gesetzt und abgefragt werden.

Die Richtung des Schattens bzw. der Blickwinkel auf das Torten-Diagramm werden in der Eigenschaft ChartPieAngle definiert.

### **ChartPieInnerRadius**

Innenradius des Torten-Diagramms

Typ [int](#)

Siehe [Liste](#), [Chart](#)

In dieser Eigenschaft kann der Innenradius des Torten-Diagramms gesetzt und ermittelt werden, sodass das Diagramm als Donut dargestellt wird. Standardmäßig beträgt der Innenradius 0.

## Kontakt

ChartPieLabelFont

Schrift der Bezeichner

Typ font

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Schrift der Bezeichner der Sektoren in einem Tortendiagramm gesetzt oder gelesen werden. Alle Bezeichner werden mit der gleichen Schrift angezeigt. Als Standard wird die Schriftart Arial verwendet.

### ChartPieShading

Schattierung des Torten-Diagramms

Typ `int`

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schattierung des Torten-Diagramms gesetzt oder ermittelt werden. Der Wert kann mit folgenden Konstanten gesetzt werden:

- `_ChartPieShadingDefault` Keine

Schattierung.

- `_ChartPieShadingFlat`

Keine Schattierung.

- `_ChartPieShadingGradientLocal`

Farbverlauf für jeden Sektor von oben nach unten heller werdend.

- `_ChartPieShadingGradientGlobal`

Farbverlauf von oben nach unten heller werdend.

- `_ChartPieShadingConcave`

Eingedellte Torten-Darstellung.

- `_ChartPieShadingRoundedEdge`

Abgerundete Sektoren-Kanten.

- `_ChartPieShadingRadial`

Radialer Farbverlauf von der Mitte des Torten-Diagramms ausgehend.

- `_ChartPieShadingRing`

Ringförmiger Farbverlauf von der Mitte des Torten-Diagramms ausgehend.

Die Konstanten können nicht miteinander kombiniert werden.

### ChartPieShadowMode

Torten-Diagramm mit Schatten

Typ logic

Siehe Liste, Chart

Mit dieser Eigenschaft wird bestimmt, ob das Torten-Diagramm mit einem Schatten (true) oder dreidimensional (false) gezeichnet wird. Die Eigenschaften ChartPieDepth und ChartPieAngle geben die Tiefe des Schattens bzw. die Höhe der Torte und den Winkel des Schattens bzw. den Betrachtungswinkel an.

## Kontakt

**ChartPieStartAngle**  Startwinkel und Anzeigereihenfolge des Tortendiagramms Typ int

Siehe [Liste](#), [Chart](#)

In dieser Eigenschaft können der Startwinkel und die Anzeigereihenfolge des Tortendiagramms gesetzt und ermittelt werden. Es können Werte zwischen -360 und +360 Grad angegeben werden. Bei negativen Werten startet der erste Abschnitt von oben um die definierte Gradzahl entgegen dem Uhrzeigersinn. Weitere Abschnitte werden der Reihenfolge nach auch entgegen dem Uhrzeigersinn erzeugt. Bei positiven Werten werden die Abschnitte im Uhrzeigersinn angeordnet. Standardmäßig beträgt der Startwinkel 0.

## Kontakt

---

**ChartPyramidElevation**  **Vertikaler Blickwinkel auf das Pyramiden-Diagramm Typ int**

Siehe [Liste](#), [Chart](#)

In dieser Eigenschaft kann der vertikale Blickwinkel (Rotation um die X-Achse) auf das Pyramiden-Diagramm gesetzt und ermittelt werden. Standardmäßig beträgt der vertikale Blickwinkel 0 Grad.

### ChartPyramidLayerGap

**Abstand der Ebenen untereinander**

**Typ** `float`

**Siehe** [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Abstand zwischen den Ebenen bei einem Pyramiden-Diagramm gesetzt werden. In dieser Eigenschaft können Werte zwischen 0.0 und 1.0 angegeben werden, die den prozentualen Anteil des Zwischenraums angeben. Standardmäßig beträgt der Abstand 0.0.



Die Gesamtsumme der Zwischenräume zwischen allen Ebenen kann 1.0 nicht überschreiten.

## Kontakt

**ChartPyramidRotation**  Horizontaler Blickwinkel auf das  
Pyramiden-Diagramm Typ int

Siehe Liste, Chart

In dieser Eigenschaft kann der horizontale Blickwinkel (Rotation um die Y-Achse) auf das Pyramiden-Diagramm gesetzt und ermittelt werden. Standardmäßig beträgt der horizontale Blickwinkel 0 Grad.

### ChartPyramidStyleData

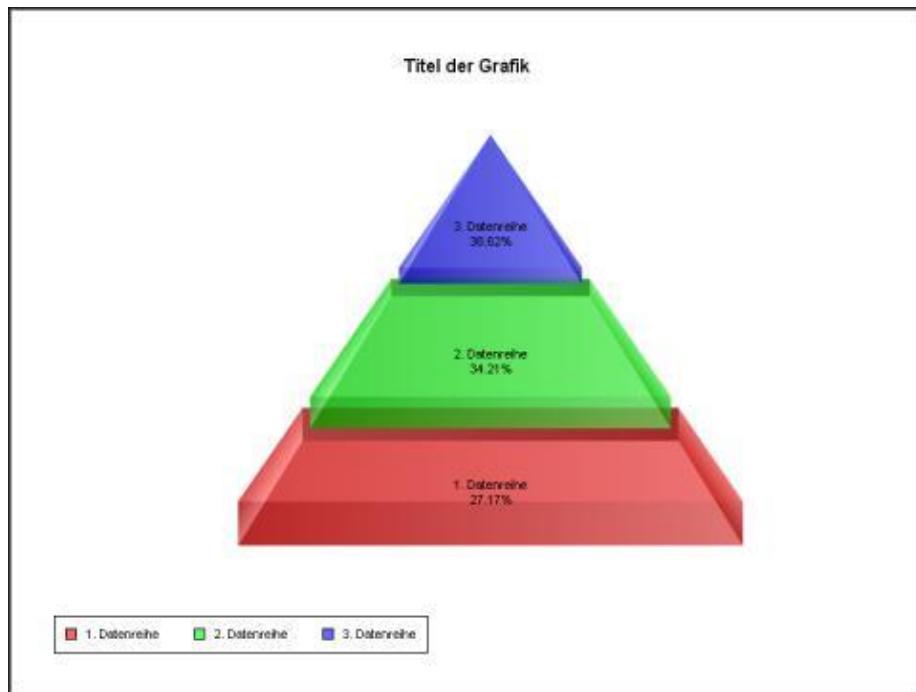
Stil der Datendarstellung

Typ `int`

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Stil der Datendarstellung in einem Pyramiden-Diagramm gesetzt und gelesen werden. Folgende Stile stehen zur Verfügung:

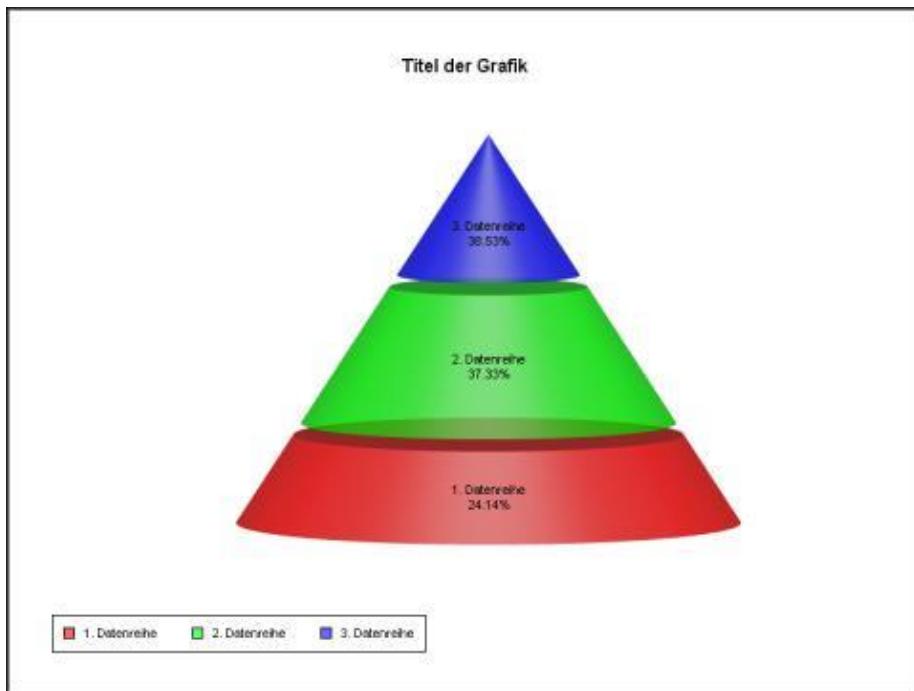
- `_ChartPyramidStyleDataDefault` / `_ChartPyramidStyleDataPyramid` Darstellung als Pyramide.



- `_ChartPyramidStyleDataCone`

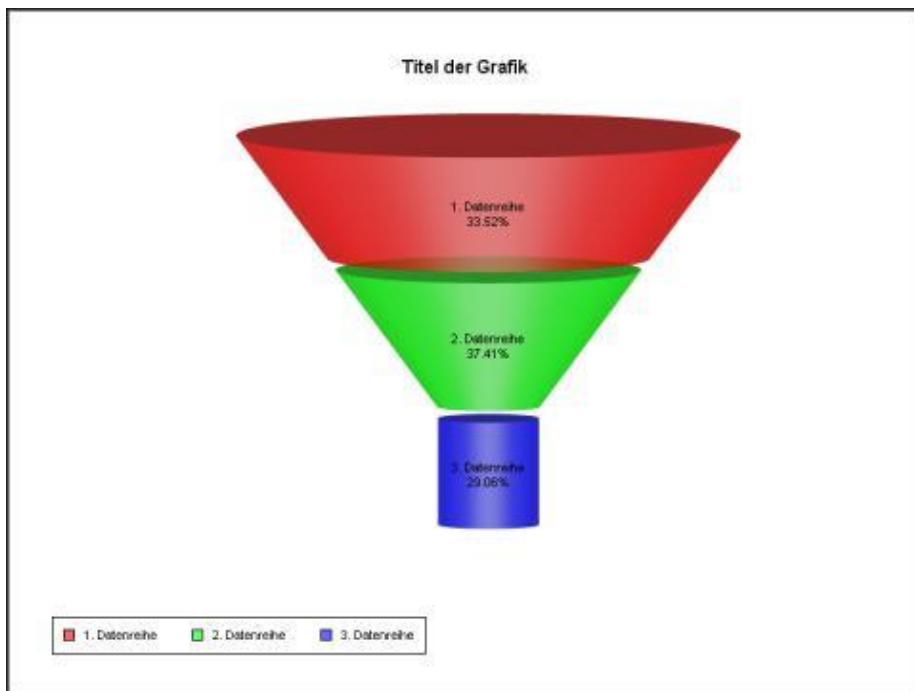
Darstellung als Kegel.

## Kontakt



- **\_ChartPyramidStyleDataFunnel**

**Darstellung als Trichter.**



## Kontakt

**ChartPyramidTubeHeight**  —

Höhe der Röhre in %

Typ **float**

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann bei der Trichterdarstellung ([ChartPyramidStyleData](#) = [ChartPyramidStyleDataFunnel](#)) in einem Pyramiden-Diagramm die Höhe der Röhre gesetzt und ermittelt werden. In dieser Eigenschaft können Werte zwischen 0.0 und 1.0 angegeben werden, die den prozentualen Anteil der Trichter-Röhre gegenüber der Gesamthöhe angeben. Standardmäßig ist der Wert 0.3.

## Kontakt

**ChartPyramidTubeRadius** —

Radius der Röhre in %

Typ **float**

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann bei der Trichterdarstellung ([ChartPyramidStyleData](#) = [ChartPyramidStyleDataFunnel](#)) in einem Pyramiden-Diagramm der Radius der Röhre gesetzt und ermittelt werden. In dieser Eigenschaft können Werte zwischen 0.0 und 1.0 angegeben werden, die den prozentualen Anteil der Trichter-Röhre gegenüber dem Radius der Einfüllöffnung angeben. Standardmäßig ist der Wert 0.2.

## Kontakt

ChartPyramidTwist  **Blickwinkel der Kamera auf das Pyramiden-Diagramm Typ int**

Siehe Liste, Chart

In dieser Eigenschaft kann der Blickwinkel der Kamera auf das Pyramiden-Diagramm gesetzt und ermittelt werden. Standardmäßig beträgt der Kamerablickwinkel 0 Grad.

## Kontakt

ChartSurfaceAmbient  Umgebungsreflektion bei einem  
Oberflächen-Diagramm Typ float

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Stärke der Umgebungsreflektion bei einem Oberflächen-Diagramm gesetzt werden. In dieser Eigenschaft können Werte zwischen 0.0 und 10.0 angegeben werden. Standardmäßig beträgt die Umgebungsreflektion 0.5.

## Kontakt

ChartSurfaceColContour  Farbe der Z-Achsenlinien  
auf der Oberfläche Typ color

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Farbe der Z-Achsenlinien eines Oberflächen-Diagramm gesetzt und gelesen werden. Ist die Eigenschaft gesetzt, werden die Achsenlinien in der gewählten Farbe auf die Oberfläche projiziert. Standardmäßig ist keine Farbe gesetzt.

## Kontakt

**ChartSurfaceColSurfaceAxis**



**Farbe der X- und Y-Achsenlinien auf der Oberfläche Typ**

color

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Farbe der X- und Y-Achsenlinien eines Oberflächen-Diagramm gesetzt und gelesen werden. Ist die Eigenschaft gesetzt, werden die Achsenlinien in der gewählten Farbe auf die Oberfläche projiziert. Standardmäßig ist keine Farbe gesetzt.

## Kontakt

**ChartSurfaceColSurfaceData**



**Farbe der Datenlinien auf der Oberfläche**

Typ [color](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Farbe der Datenlinien eines Oberflächen-Diagramm gesetzt und gelesen werden. Ist die Eigenschaft gesetzt, werden Verbindungslinien zwischen den Datenpunkten in der gewählten Farbe auf die Oberfläche projiziert. Standardmäßig ist keine Farbe gesetzt.

## Kontakt

ChartSurfaceContinuous

Fließende Farbübergänge Typ

logic

Siehe Liste, Chart

Mit dieser Eigenschaft kann bei Oberflächen-Diagrammen gesetzt oder ermittelt werden, ob fließende Farbübergänge verwendet werden sollen, oder nicht. Standardmäßig werden keine fließenden Farbübergänge verwendet.

## Kontakt

---

ChartSurfaceDiffuse  Streureflektion bei einem  
Oberflächen-Diagramm Typ float

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Stärke der Streureflektion bei einem Oberflächen-Diagramm gesetzt werden. In dieser Eigenschaft können Werte zwischen 0.0 und 10.0 angegeben werden. Standardmäßig beträgt die Streureflektion 0.5.

## Kontakt

ChartSurfaceElevation  Vertikaler Blickwinkel auf das  
Oberflächen-Diagramm Typ int

Siehe Liste, Chart

In dieser Eigenschaft kann der vertikale Blickwinkel (Rotation um die X-Achse) auf das Oberflächen-Diagramm gesetzt und ermittelt werden. Standardmäßig beträgt der vertikale Blickwinkel 30 Grad.

## Kontakt

**ChartSurfaceLegendLength**  Länge der Farb-Legende beim  
Oberflächen-Diagramm Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Länge der Farb-Legende bei Oberflächen-Diagrammen gesetzt oder ermittelt werden. Die Angabe erfolgt in Pixel. Standardmäßig ist die Länge auf 200 Pixel gesetzt.

## Kontakt

---

**ChartSurfacePerspective**

**Stärke des Perspektiveneffekts**

**Typ int**

**Siehe Liste, Chart**

In dieser Eigenschaft kann die Stärke des Perspektiveneffekts des Oberflächen-Diagramm gesetzt und ermittelt werden. Es können Werte zwischen 0 und 100 angegeben werden. Standardmäßig beträgt der Perspektiveneffekt 12.

Je stärker der Perspektiveneffekt ist, desto stärker ist der Größenunterschied zwischen zwei Objekten, die gleich hoch, aber unterschiedlich weit entfernt sind.

## Kontakt

ChartSurfacePlotHeight  Höhe der Z-Achse beim  
Oberflächen-Diagramm Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Höhe der Z-Achse bei Oberflächen-Diagrammen gesetzt oder ermittelt werden. Die Angabe erfolgt in Pixel. Standardmäßig ist die Höhe auf 200 Pixel gesetzt.

## Kontakt

ChartSurfaceRotation  Horizontaler Blickwinkel auf das  
Oberflächen-Diagramm Typ int

Siehe Liste, Chart

In dieser Eigenschaft kann der horizontale Blickwinkel (Rotation um die Y-Achse) auf das Oberflächen-Diagramm gesetzt und ermittelt werden. Standardmäßig beträgt der horizontale Blickwinkel 45 Grad.

## Kontakt

**ChartSurfaceSamplesX**  $\sqcup$  Datenanzahl der X-Achse  
für Interpolation Typ int

Siehe [Liste](#), [Chart](#), [ChartSurfaceSamplesY](#),

Mit dieser Eigenschaft kann die Datenanzahl der X-Achse für Interpolation bei Oberflächen-Diagrammen gesetzt oder ermittelt werden. Standardmäßig werden keine Daten interpoliert.



Je mehr Punkte interpoliert werden sollen, umso länger dauert die Erstellung des Diagramms, jedoch wird die Darstellung bei mehr Punkten verfeinert. Ist die Eigenschaft [ChartSurfaceSmooth](#) auf true gesetzt, genügen meist relativ wenige Punkte.

## Kontakt

ChartSurfaceSamplesY  Datenanzahl der Y-Achse

für Interpolation Typ int

Siehe [Liste](#), [Chart](#), [ChartSurfaceSamplesX](#),

Mit dieser Eigenschaft kann die Datenanzahl der Y-Achse für Interpolation bei Oberflächen-Diagrammen gesetzt oder ermittelt werden. Standardmäßig steht diese Eigenschaft auf dem Wert -1, bei dem die Datenanzahl der X-Achse verwendet wird.



Je mehr Punkte interpoliert werden sollen, umso länger dauert die Erstellung des Diagramms, jedoch wird die Darstellung bei mehr Punkten verfeinert. Ist die Eigenschaft ChartSurfaceSmooth auf true gesetzt, genügen meist relativ wenige Punkte.

## Kontakt

**ChartSurfaceShadingMode**  Untergrund des  
Oberflächen-Diagramms Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft kann der Untergrund in einem Oberflächen-Diagramm gesetzt und gelesen werden. Folgende Untergründe stehen zur Verfügung:

- \_ChartSurfaceShadingDefault / \_ChartSurfaceShadingSmooth Untergrund mit Kantenglättung.
- \_ChartSurfaceShadingTriangle  
Untergrund ist aus Dreiecken zusammengesetzt.
- \_ChartSurfaceShadingRectangle  
Untergrund ist aus Rechtecken zusammengesetzt.
- \_ChartSurfaceShadingTriangleFrame  
Untergrund ist aus Dreiecken zusammengesetzt. Es werden nur die Ränder angezeigt. Die Dicke der Ränder kann in der Eigenschaft ChartSurfaceWireWidth definiert werden.
- \_ChartSurfaceShadingRectangleFrame  
Untergrund ist aus Rechtecken zusammengesetzt. Es werden nur die Ränder angezeigt. Die Dicke der Ränder kann in der Eigenschaft ChartSurfaceWireWidth definiert werden.

## Kontakt

---

**ChartSurfaceShine**

Glanz bei einem Oberflächen-Diagramm

Typ **float**

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Stärke des Glanzes bei einem Oberflächen-Diagramm gesetzt werden.

In dieser Eigenschaft können Werte zwischen 0.0 und 100.0 angegeben werden. Standardmäßig beträgt die Stärke des Glanzes 8.0.

### ChartSurfaceSmooth

#### Interpolation mit Kantenglättung

Typ logic

Liste, Chart,

Siehe ChartSurfaceSamplesX,

ChartSurfaceSamplesY

Mit dieser Eigenschaft kann bei Oberflächen-Diagrammen gesetzt oder ermittelt werden, ob die Kanten bei einer Interpolation geglättet (true) werden sollen, oder nicht. Standardmäßig werden die Kanten geglättet.

Ist diese Eigenschaft auf true gesetzt, können oftmals mit wenigen Punkten zur Interpolation ähnliche Ergebnisse erreicht werden, wie mit vielen Punkten. Dies ist möglich, da die Punkte mit einer Spline-Linie verbunden werden, statt mit einer geraden.

## Kontakt

---

ChartSurfaceSpecular  Spiegelreflektion bei einem  
Oberflächen-Diagramm Typ float

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Stärke der Spiegelreflektion bei einem Oberflächen-Diagramm gesetzt werden. In dieser Eigenschaft können Werte zwischen 0.0 und 20.0 angegeben werden. Standardmäßig beträgt die Spiegelreflektion 1.0.

## Kontakt

---

### ChartSurfaceTitleFontX

Schrift der x-Achsen-Beschriftung

Typ [font](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schriftart, mit der der Titel der x-Achse gezeichnet wird, gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet.

## Kontakt

---

### ChartSurfaceTitleFontY

Schrift der y-Achsen-Beschriftung

Typ [font](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schriftart, mit der der Titel der y-Achse gezeichnet wird, gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet.

## Kontakt

---

### ChartSurfaceTitleFontZ

Schrift der z-Achsen-Beschriftung

Typ [font](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schriftart, mit der der Titel der z-Achse gezeichnet wird, gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet.

## Kontakt

**ChartSurfaceTitleX**



**Beschriftung der linken unteren Achse**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste](#), [Chart](#)

**Mit dieser Eigenschaft kann die Beschriftung der linken unteren Achse eines Oberflächen-Diagramms gesetzt und gelesen werden. Standardmäßig ist diese Eigenschaft nicht gesetzt.**

## Kontakt

**ChartSurfaceTitleY**



**Beschriftung der rechten unteren Achse**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste](#), [Chart](#)

**Mit dieser Eigenschaft kann die Beschriftung der rechten unteren Achse eines Oberflächen-Diagramms gesetzt und gelesen werden. Standardmäßig ist diese Eigenschaft nicht gesetzt.**

## Kontakt

ChartSurfaceTitleZ



Beschriftung der vertikalen Achse

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Beschriftung der vertikalen Achse eines Oberflächen-Diagramms gesetzt und gelesen werden. Standardmäßig ist diese Eigenschaft nicht gesetzt.

## Kontakt

ChartSurfaceTwist  **Blickwinkel der Kamera auf das Oberflächen-Diagramm Typ int**

Siehe [Liste](#), [Chart](#)

In dieser Eigenschaft kann der Blickwinkel der Kamera auf das Oberflächen-Diagramm gesetzt und ermittelt werden. Standardmäßig beträgt der Kamerablickwinkel 0 Grad.

## Kontakt

**ChartSurfaceWireWidth** —

Dicke der Ränder

Typ [int](#)

Siehe [Liste](#), [Chart](#)

In dieser Eigenschaft kann bei der Ränderdarstellung ([ChartSurfaceShadingMode](#) = [ChartSurfaceShadingTriangleFrame](#) bzw. [ChartSurfaceShadingMode](#) = [ChartSurfaceShadingRectangleFrame](#)) von Oberflächen-Diagrammen die Dicke der Ränder gesetzt und ermittelt werden. Die Dicke wird in Pixel angegeben.  
Standardmäßig beträgt die Dicke 1 Pixel.

## Kontakt

### ChartTitleArea

Position des Titels

Typ rect

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Position des Titels in einem Chart-Objekt gesetzt und abgefragt werden.

Ist beim Setzen des übergebenen Rechtecks die Komponente right kleiner oder gleich der Komponente left und die Komponente bottom kleiner oder gleiche der Komponente top, dann wird die Position des Titels verschoben, die Größe wird beibehalten.

Der Text des Titels wird in der Eigenschaft ChartTitleText angegeben. Der Text wird horizontal zentriert und falls die Breite des Bereichs nicht ausreicht, umgebrochen. Nach unten wird über den definierten Bereich hinaus geschrieben.

Beispiele:

```
tChart->spChartTitleArea # RectMake(300, 10, 500, 30); // define title position
```

## Kontakt

---

**ChartTitleColBkg**

Hintergrundfarbe des Titels

Typ color

Siehe Liste, Chart,

**Siehe** ChartTitleColFg

Mit dieser Eigenschaft kann die Hintergrundfarbe des Titels gesetzt und abgefragt werden.

## Kontakt

---

### ChartTitleColFg

Vordergrundfarbe des Titels

Typ [color](#)

Siehe [Liste](#), [Chart](#),

Siehe [ChartTitleColBkg](#)

Mit dieser Eigenschaft kann die Vordergrundfarbe des Titels des [Chart](#)-Objekts gesetzt und abgefragt werden.

## Kontakt

**ChartTitleFont**

Schriftart des Titels

Typ font

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Schriftart des Titels des Chart-Objekts gesetzt und abgefragt werden. Als Standard wird die Schriftart Arial verwendet.

## Kontakt

ChartTitleText

**Titel der Grafik**

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Chart](#)

In dieser Eigenschaft kann der Titel der Grafik angegeben oder abgefragt werden. Der Titel wird zentriert über der Grafik innerhalb des Ausgabebereiches angezeigt. Der Titel kann ebenfalls bei der Anweisung [ChartOpen\(\)](#) angegeben werden. Eine leere Zeichenkette löscht den Titel.

## Kontakt

**ChartWidth**



Breite des Ausgabebereichs des Chart-Objekts

Typ int

Siehe Liste, Chart, ChartHeight, ChartOpen()

Mit dieser Eigenschaft kann die Breite des Ausgabebereiches des Chart-Objekts gesetzt oder ermittelt werden.

## Kontakt

**ChartXYAxisColY**



**Farbe der Y-Achse**

**Typ color**

Liste, Chart,

ChartXYAxisY,

Siehe ChartXYAxisOffsetY,

ChartXYAxisTitleY,

ChartXYAxisTitleAlignY

Die Eigenschaft definiert die Farbe der Achse inklusive Achsenbeschriftung. Die Eigenschaft wirkt sich bei allen Ausprägungen von ChartXYAxisY aus.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen.

Somit kann die Farbe für jede Achse separat generiert werden.

Über die Eigenschaften ChartXYAxisOffsetY, ChartXYAxisTitleY und ChartXYAxisTitleAlignY können die Position und das Aussehen der Achse verändert werden.

### ChartXYAxisOffsetY

Abstand der Y-Achse zum Diagramm

Typ int

Liste, Chart, ChartXYAxisY,

Siehe ChartXYAxisColY,

ChartXYAxisTitleY,

ChartXYAxisTitleAlignY

Die Eigenschaft definiert den Abstand der Achse zum Diagramm. Positive Werte verschieben die Achse vom Diagramm weg, negative Werte zum Diagramm hin.



Die Eigenschaft hat keine Auswirkung, wenn die Eigenschaft ChartXYAxisY den Wert ChartXYAxisDefault hat.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Somit kann für jede Achse der Abstand separat definiert werden.

Über die Eigenschaften ChartXYAxisColY, ChartXYAxisTitleY und ChartXYAxisTitleAlignY kann das Aussehen der Achse verändert werden.

## Kontakt

**ChartXYAxisTitleAlignY**



Position des Titels der Y-Achse der Datenreihe Typ int

Liste, Chart, ChartXYAxisY,

Siehe ChartXYAxisOffsetY, ChartXYAxisColY,

ChartXYAxisTitleY

Die Eigenschaft definiert Position und Richtung des Titels (Eigenschaft

ChartXYAxisTitleY) der Achse. Die Eigenschaft wirkt sich bei den Ausprägungen

ChartXYAxisLeft und ChartXYAxisRight von ChartXYAxisY aus. Folgende Werte sind erlaubt:

- \_ChartAlignBottomLeft (1)

Der Text befindet sich links unter der Achse und wird vertikal dargestellt.

- \_ChartAlignBottomCenter (2)

Der Text befindet sich links unter der Achse und wird vertikal dargestellt.

- \_ChartAlignBottomRight (3)

Der Text befindet sich rechts unter der Achse und wird vertikal dargestellt.

- \_ChartAlignLeft (4)

Der Titel befindet sich links von der Achse, vertikal zentriert und wird vertikal dargestellt.

- \_ChartAlignCenter (5)

Der Titel befindet sich auf der Achse, vertikal zentriert und wird vertikal dargestellt.

- \_ChartAlignRight (6)

Der Titel befindet sich rechts von der Achse, vertikal zentriert und wird vertikal dargestellt.

- \_ChartAlignTopLeft (7)

Der Titel befindet sich links über der Achse und wird vertikal dargestellt.

- \_ChartAlignTopCenter (8)

Der Titel befindet sich horizontal zentriert über der Achse.

- \_ChartAlignTopRight (9)

Der Titel befindet sich rechts über der Achse und wird vertikal dargestellt.

- \_ChartAlignTopLeft2 (10)

Der Text befindet sich links über der Achse.

- \_ChartAlignTopRight2 (11)

Der Text befindet sich rechts über der Achse.

- \_ChartAlignBottomLeft2 (12)

Der Text befindet sich links unter der Achse.

- \_ChartAlignBottomRight2 (13)

## Kontakt

Der Text befindet sich rechts unter der Achse.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen.  
Somit kann die Position des Titels für jede Y-Achse separat definiert werden.

Über die Eigenschaften ChartXYAxisOffsetY, ChartXYAxisColY und ChartXYAxisTitleY können die Position und das Aussehen der Achse verändert werden.

### **ChartXYAxisTitleY**

**Titel der Y-Achse der Datenreihe**

**Typ** alpha

Liste, Chart, ChartXYAxisY,

**Siehe** ChartXYAxisOffsetY,

ChartXYAxisColY,

ChartXYAxisTitleAlignY

Die Eigenschaft definiert den Titel für die Achse. Die Beschriftung wird in Abhängigkeit von der Eigenschaft ChartXYAxisTitleAlignY positioniert. Die Eigenschaft wirkt sich bei den Ausprägungen ChartXYAxisLeft und ChartXYAxisRight von ChartXYAxisY aus.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen.

Somit kann für jede Y-Achse ein separater Titel definiert werden.

Über die Eigenschaften ChartXYAxisOffsetY, ChartXYAxisColY und ChartXYAxisTitleAlignY können die Position und das Aussehen der Achse verändert werden.

## Kontakt

ChartXYAxisY Y-Achse für

Datenmenge Typ int

Liste, Chart,  
ChartXYAxisOffsetY,  
Siehe ChartXYAxisColY,  
ChartXYAxisTitleY,  
ChartXYAxisTitleAlignY

Über die Eigenschaft kann gesteuert werden, ob der Datenmenge eine eigene Achse zugeordnet werden soll, und wenn ja, ob diese links oder rechts vom Diagramm gezeichnet werden soll.

Folgende Konstanten können angegeben werden:

- \_ChartXYAxisDefault (0)

Es wird keine neue Achse generiert.

- \_ChartXYAxisLeft (1)

Es wird eine neue Achse links vom Diagramm generiert.

- \_ChartXYAxisRight (2)

Es wird eine neue Achse rechts vom Diagramm generiert.

Die Konstanten können nicht miteinander kombiniert werden.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen.

Somit kann für jede Datenreihe eine separate Achse generiert werden.

Bei den Konstanten \_ChartXYAxisLeft und \_ChartXYAxisRight können über die Eigenschaften ChartXYAxisOffsetY, ChartXYAxisColY, ChartXYAxisTitleY und ChartXYAxisTitleAlignY die Position und das Aussehen der Achse verändert werden. Bei der Konstanten \_ChartXYAxisDefault wird über ChartXYTitleAlignY die Position der Hauptachse definiert.



Wird bei einer Datenreihe eine zusätzliche Achsenbeschriftung eingefügt (ChartXYAxisY != \_ChartXYAxisDefault) und die Eigenschaft ChartXYSwapXY gesetzt, wird die Datenreihe unter Umständen falsch skaliert. Dies liegt daran, dass die Koordinatendiagramme nur eine X-Achse haben können.

### ChartXYBarGap

**Abstand der Balken untereinander**

**Typ** `float`

**Siehe** [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Abstand zwischen den Balken einer Datenreihe bei einem Koordinaten-Diagramm gesetzt werden. Die Breite der einzelnen Balken wird aus der Breite des Bildes ([ChartWidth](#)) und der Anzahl der Werte in einer Datenreihe bestimmt. In dieser Eigenschaft können Werte zwischen 0.0 und 1.0 angegeben werden, die den prozentualen Anteil des Zwischenraums angeben.

Der voreingestellte Wert ist 0.25. Das bedeutet, dass 12,5% der Gesamtbreite des Balkens links und 12,5% der Gesamtbreite des rechts von dem Balken frei gelassen werden. Bei der Angabe von 1.0 in dieser Eigenschaft werden die Balken als senkrechte Striche dargestellt. Bei der Angabe von 0.0 werden keine Zwischenräume gezeichnet.

Die Eigenschaft wird beim Schließen des [ChartData](#)-Objekts in das [Chart](#)-Objekt übertragen. Der Abstand zwischen den Balken kann somit für jede Datenreihe separat gesetzt werden.

### ChartXYBarShading

---

Schattierung der Balken

Typ `int`

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schattierung der Balken in einem Koordinaten-Diagramm gesetzt und gelesen werden. Für die Schattierung stehen folgende Konstanten zur Verfügung:

- `_ChartXYBarShadingDefault` Ohne

Schattierung.

- `_ChartXYBarShadingSoftTop`

Weicher Farbverlauf hell-dunkel von oben nach unten.

- `_ChartXYBarShadingSoftLeft`

Weicher Farbverlauf hell-dunkel von links nach rechts.

- `_ChartXYBarShadingSoftRight`

Weicher Farbverlauf hell-dunkel von rechts nach links.

- `_ChartXYBarShadingSoftBottom`

Weicher Farbverlauf hell-dunkel von unten nach oben.

- `_ChartXYBarShadingGlassTop`

Farbverlauf mit Glaseffekt von oben nach unten.

- `_ChartXYBarShadingGlassLeft`

Farbverlauf mit Glaseffekt von links nach rechts.

- `_ChartXYBarShadingGlassRight`

Farbverlauf mit Glaseffekt von rechts nach links.

- `_ChartXYBarShadingGlassBottom` Farbverlauf mit

Glaseffekt von unten nach oben.

- `_ChartXYBarShadingGradientTop`

Farbverlauf von oben nach unten.

- `_ChartXYBarShadingGradientBottom`

Farbverlauf von unten nach oben.

Die Konstanten können nicht miteinander kombiniert werden.

### ChartXYBarShape

Form der Balken

Typ int

Siehe Liste, Chart

Änderungen in dieser Eigenschaft wirken sich nur aus, wenn die Eigenschaft

ChartXYStyleData auf ChartXYStyleDataBar gesetzt ist.

Mit dieser Eigenschaft kann die Form, in der der Balken in einem Koordinaten-Diagramm gezeichnet wird, gesetzt und ermittelt werden. Folgende Darstellungen stehen zur Verfügung:

- \_ChartXYBarShapeDefault  
Balkenform
- \_ChartXYBarShapeDiamond  
Rautenform
- \_ChartXYBarShapeTriangleUp Nach  
oben zeigendes Dreieck
- \_ChartXYBarShapeTriangleRight Nach  
rechts zeigendes Dreieck
- \_ChartXYBarShapeTriangleLeft Nach  
links zeigendes Dreieck
- \_ChartXYBarShapeTriangleDown Nach  
unten zeigendes Dreieck
- \_ChartXYBarShapeCircle  
Kreisform

Die Konstanten können nicht miteinander kombiniert werden.

Die Form der Balken wird beim Schließen der Datenreihe mit der Anweisung ChartDataClose() an das Chart-Objekt übertragen. Für jede Datenreihe kann eine andere Form gewählt werden.

### **ChartXYColBkg**

Hintergrundfarbe der Grafik

Typ color

Liste, Chart,

Siehe ChartXYColBkgAlt

Mit dieser Eigenschaft kann die Füllfarbe für den Hintergrund eines Koordinaten-Diagramms gesetzt und gelesen werden. Die Farbe wird für die ungeraden Abschnitte der y-Achse verwendet. Die geraden Abschnitte werden in der Farbe, die in der Eigenschaft ChartXYColBkgAlt angegeben ist, angezeigt.

## Kontakt

---

**ChartXYColBkgAlt**

Hintergrundfarbe der Grafik

Typ color

Liste, Chart,

Siehe

Mit dieser Eigenschaft kann die Füllfarbe für den Hintergrund eines Koordinaten-Diagramms gesetzt und gelesen werden. Die Farbe wird für die geraden Abschnitte der y-Achse verwendet. Die ungeraden Abschnitte werden in der Farbe, die in der Eigenschaft ChartXYColBkg angegeben ist, angezeigt.

## Kontakt

**ChartXYColBorder**



**Rahmenfarbe der Balken / Linien / Flächen**

Typ [color](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Farbe des Rahmens der Balken, Linien und Flächen bei Koordinaten-Diagrammen gesetzt und gelesen werden.

Die Farbe wird beim Schließen des [ChartData](#)-Objekts in das [Chart](#)-Objekt übertragen. Sie kann somit für jede Datenreihe separat gesetzt werden, wenn die Änderung vor der Anweisung [ChartDataClose\(\)](#) erfolgt.

### ChartXYColData

**Farbe der Balken oder Flächen**

Typ [color](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Farbe der Balken oder der Flächen in einem Koordinaten-Diagramm gesetzt und gelesen werden. Die Eigenschaft wirkt sich nur aus, wenn beim Erzeugen des [ChartData](#)-Objekts mit [ChartDataOpen\(\)](#) keine Farbwerte mit angegeben wurden.

Die Eigenschaft wird beim Schließen des [ChartData](#)-Objekts in das [Chart](#)-Objekt übertragen.

Die Farbe der Balken kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

**ChartXYColGridX**



**Farbe der Gitternetzlinien der X-Achse**

**Typ** color

**Siehe** Liste, Chart, ChartXYColGridY

Mit dieser Eigenschaft kann die Farbe der Gitternetzlinien der X-Achse in einem Koordinaten-Diagramm gesetzt und gelesen werden. Standardmäßig ist die Farbe der Gitternetzlinien der X-Achse WinColTransparent.

## Kontakt

**ChartXYColGridY**



**Farbe der Gitternetzlinien der Y-Achse**

**Typ** color

**Siehe** Liste, Chart, ChartXYColGridX

Mit dieser Eigenschaft kann die Farbe der Gitternetzlinien der Y-Achse in einem Koordinaten-Diagramm gesetzt und gelesen werden. Standardmäßig ist die Farbe der Gitternetzlinien der Y-Achse WinColLightGray.

## Kontakt

**ChartXYColTrend**

**Farbe der Trendlinie**

**Typ** color

**Siehe** Liste, Chart

Mit dieser Eigenschaft kann die Farbe der Trendlinie in einem Koordinaten-Diagramm gesetzt und gelesen werden. Die Eigenschaft wirkt sich nur aus, wenn die Eigenschaft ChartXYTrendType ungleich ChartXYTrendTypeDefault bzw.

ChartXYTrendTypeNone ist. Standardmäßig wird die Farbe der Trendlinie automatisch ermittelt.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Die Farbe der Trendlinie kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

**ChartXYDepth**



**Tiefe des Balkens / der Linie / der Fläche**

**Typ** int

**Siehe** Liste, Chart

Mit dieser Eigenschaft kann die Tiefe der Balken, der Linien und der Flächen in einem Koordinaten-Diagramm gesetzt und gelesen werden. Wird eine Tiefe von 0 angegeben, erfolgt die Ausgabe zweidimensional.

## Kontakt

---

### ChartXYDepthGap

Abstand der Balken in der Tiefe

Typ [int](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Abstand der Balken von verschiedenen Datenreihen in einem Koordinaten-Diagramm gesetzt und gelesen werden. Der Abstand wird in Pixeln angegeben.

Die Eigenschaft wird beim Schließen des [ChartData](#)-Objekts in das [Chart](#)-Objekt übertragen. Der Abstand zur nächsten Datenreihen kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

---

### ChartXYGapDash

Linienart bei nicht vorhandenen Werten

Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Linienart bei nicht vorhandenen Werten in einem Koordinaten-Diagramm gesetzt und gelesen werden. Für die Linienart stehen folgende Konstanten zur Verfügung:

- \_ChartXYGapDashDefault / \_ChartXYGapDashNone Keine

Linie.

- \_ChartXYGapDashLine

Fortlaufende Linie. ( \_\_\_\_\_ )

- \_ChartXYGapDashDashLine

Gestrichelte Linie. ( - - - - - )

- \_ChartXYGapDashDotLine

Gepunktete Linie. ( ..... )

- \_ChartXYGapDashDotDashLine

Strichpunkt Linie. ( - - - - . )

- \_ChartXYGapDashAltDashLine

Alternativ gestrichelte Linie. ( - - - - - - )

Die Konstanten können nicht miteinander kombiniert werden.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Die Linienart für nicht vorhandene Werte kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

---

### **ChartXYLabelAngleX**

**Winkel der Beschriftung der x-Achse**

**Typ** `float`

**Siehe** [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Winkel der Bezeichner auf der x-Achse eines Koordinaten-Diagramms angegeben werden. Alle Beschriftungen werden im gleichen Winkel angezeigt. Die Drehung erfolgt bei positiven Zahlen gegen, und bei negativen Zahlen im Uhrzeigersinn.

## Kontakt

**ChartXYLabelColData**



Textfarbe für Beschriftung der Werte und zusätzlicher Daten

Typ [color](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft wird die Textfarbe für die Beschriftungen des eigentlichen Wertes ([ChartXYStyleLabelData](#)) sowie der zusätzlichen Daten ([ChartXYStyleLabelDataExtra](#)) definiert.

**ChartXYLabelColSum**

Textfarbe für Summen-Beschriftung

Typ color

Siehe Liste, Chart

Mit dieser Eigenschaft wird die Textfarbe für die Summen-Beschriftungen (ChartXYStyleLabelSum) definiert.

## Kontakt

**ChartXYLabelFontData**



**Schriftart der Beschriftung der Daten**

**Typ** font

**Siehe** Liste, Chart, ChartXYStyleLabel

Mit dieser Eigenschaft kann die Schriftart der Beschriftung der Daten eines Koordinaten-Diagramms gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet. Die Beschriftung kann mit der Eigenschaft ChartXYStyleLabel gesetzt werden.

## Kontakt

**ChartXYLabelFontX**



Schrift für die Bezeichner auf der x-Achse

Typ font

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Schrift für die Bezeichner auf der x-Achse des Koordinaten-Diagramms gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet.

## Kontakt

---

### ChartXYLabelFontY

Schrift der Einteilung auf der y-Achse

Typ font

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Schrift der Einteilung der y-Achse eines Koordinaten-Diagramms gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet.

## Kontakt

ChartXYLabelRotData Rotationswinkel

für Beschriftungen Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft wird der Rotationswinkel des eigentlichen Wertes (ChartXYStyleLabelData) sowie der zusätzlichen Daten (ChartXYStyleLabelDataExtra) definiert. Die Drehung erfolgt bei positiven Zahlen gegen, und bei negativen Zahlen im Uhrzeigersinn.

## Kontakt

**ChartXYLegendText** 

**Text in der Legende**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Text in der Legende für eine Datenreihe in einem Koordinaten-Diagramm gesetzt und gelesen werden. Die Eigenschaft wird beim Schließen des [ChartData](#)-Objekts in das [Chart](#)-Objekt übertragen. Der Text der Legende muss für jede Datenreihe separat gesetzt werden.

**ChartXYLineSymbol**



Symbol auf der Linie

Typ int

Liste, Chart,

Siehe ChartXYLineStyleSize,

ChartXYLineSymbolParam

Mit dieser Eigenschaft kann bei der Liniendarstellung (ChartXYStyleData = ChartXYStyleDataLine) in einem Koordinaten-Diagramm ein Symbol für jeden Wert gesetzt oder gelesen werden. Es werden nur dann Symbole auf die Linie gezeichnet, wenn die Eigenschaft ChartXYDepth den Wert 0 hat.

Standardmäßig werden keine Symbole auf die Linie gezeichnet. Folgende Symbole stehen zur Verfügung:

- \_ChartSymbolNone

Standardwert. Es wird kein Symbol gezeichnet.

- \_ChartSymbolSquare

Rechteck.

- \_ChartSymbolDiamond

Raute.

- \_ChartSymbolTriangleUp

Dreieck nach oben zeigend.

- \_ChartSymbolTriangleLeft

Dreieck nach links zeigend.

- \_ChartSymbolTriangleRight

Dreieck nach rechts zeigend.

- \_ChartSymbolTriangleDown

Dreieck nach unten zeigend.

- \_ChartSymbolCircle

Kreis.

- \_ChartSymbolSphereGlass Kugel

mit Glaseffekt.

- \_ChartSymbolSphereBright

Hellere Kugel.

- \_ChartSymbolSphereSolid Kugel.

- \_ChartSymbolCross

Kreuz. Der prozentuale Anteil der Arme wird in der Eigenschaft ChartXYLineSymbolParam definiert.

## Kontakt

- \_ChartSymbolCrossDiag

**Diagonales Kreuz.** Der prozentuale Anteil der Arme wird in der Eigenschaft ChartXYLineSymbolParam definiert.

- \_ChartSymbolPolygon

**Polygon.** Die Anzahl der Seiten wird in der Eigenschaft ChartXYLineSymbolParam definiert.

- \_ChartSymbolStar

**Polygon.** Die Anzahl der Ecken wird in der Eigenschaft ChartXYLineSymbolParam definiert.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Die Darstellung kann somit für jede Datenreihe separat gesetzt werden.

### **ChartXYLineSymbolParam**

Parameter des Liniensymbols

Typ int

Siehe Liste, Chart,

Siehe ChartXYLineSymbol

Mit dieser Eigenschaft kann der Parameter eines Liniensymbols in einem Koordinaten-Diagramm gesetzt oder gelesen werden. Das Symbol wird in der Eigenschaft ChartXYLineSymbol angegeben. Es können Werte zwischen 0 und 100 angegeben werden. Standardmäßig ist der Wert 3.

Diese Eigenschaft wirkt sich auf folgende Symbole aus:

- ChartSymbolCross

Prozentualer Anteil der Arme des Kreuzes.

- ChartSymbolCrossDiag

Prozentualer Anteil der Arme des Kreuzes.

- ChartSymbolPolygon

Anzahl der Seiten des Polygons.

- ChartSymbolStar

Anzahl der Ecken des Sterns.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen.

Der Parameter des Symbols kann somit für jede Datenreihe separat gesetzt werden.

### **ChartXYLineSymbolSize**

**Höhe und Breite des Liniensymbols**

**Typ** int

**Siehe** ChartXYLineSymbol

Mit dieser Eigenschaft kann die Höhe und Breite von Liniensymbolen in einem Koordinaten-Diagramm gesetzt oder gelesen werden. Das Symbol wird in der Eigenschaft ChartXYLineSymbol angegeben.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Die Größe des Symbols kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

**ChartXYLineWidth** |

Linienbreite

Typ [int](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Breite von Linien in der Liniendarstellung

([ChartXYStyleData](#) = [ChartXYStyleDataLine](#)) in einem Koordinaten-Diagramm gesetzt und gelesen werden.

Die Eigenschaft wird beim Schließen des [ChartData](#)-Objekts in das [Chart](#)-Objekt übertragen. Die Linienbreite kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

**ChartXYMinTickIncY**

Schrittweite der y-Achse

Typ [float](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schrittweite der Beschriftung auf der y-Achse eines Koordinaten-Diagramms gesetzt und ausgelesen werden.

## Kontakt

### ChartXYStyleData

Stil der Datendarstellung

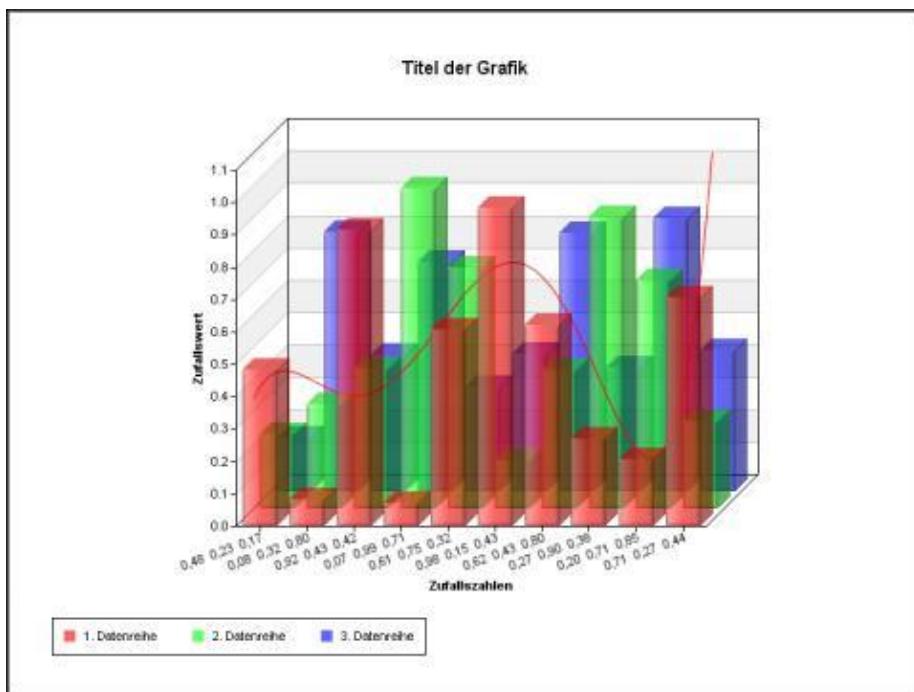
Typ `int`

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann der Stil der Datendarstellung in einem Koordinaten-Diagramm gesetzt und gelesen werden. Folgende Stile stehen zur Verfügung:

- `_ChartXYStyleDataDefault / _ChartXYStyleDataBar`

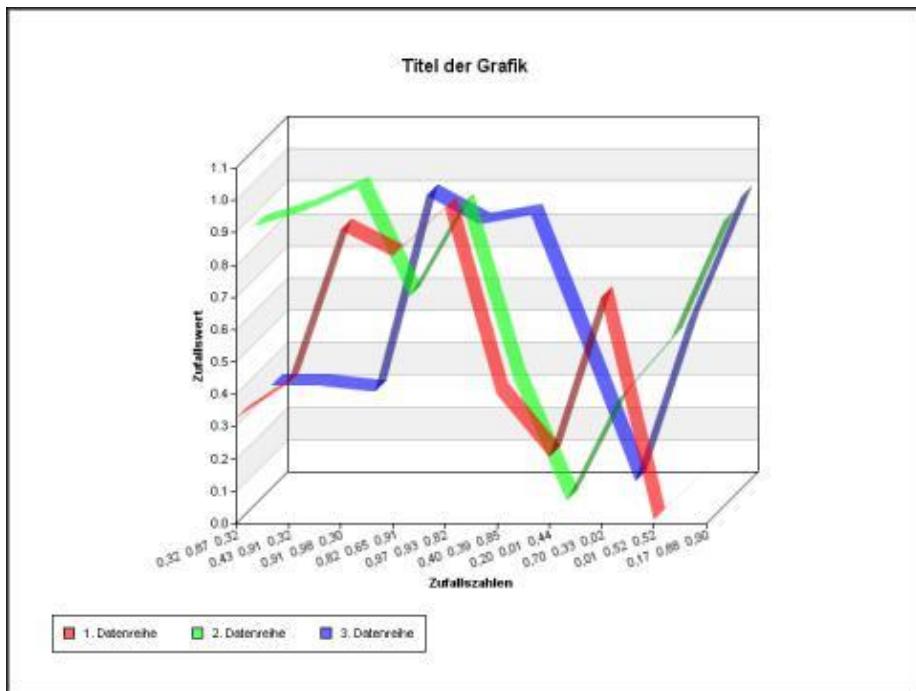
Darstellung als Balken. Die Form der Balken kann in der Eigenschaft [ChartXYBarShape](#) weiter angepasst werden.



- `_ChartXYStyleDataLine`

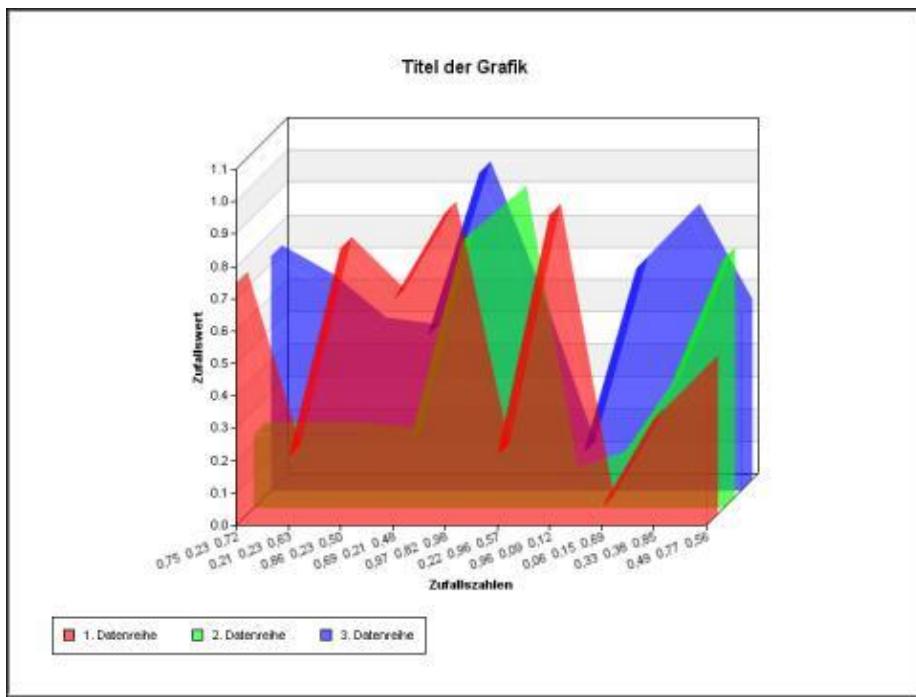
Darstellung als Linie.

## Kontakt



- **\_ChartXYStyleDataArea**

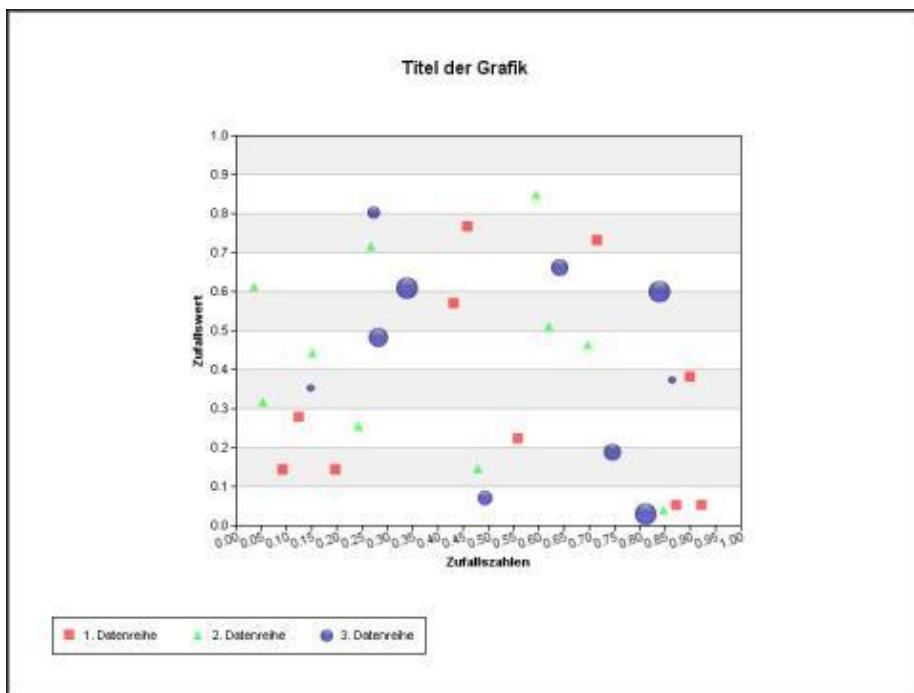
**Darstellung als Fläche.**



- **\_ChartXYStyleDataScatter**

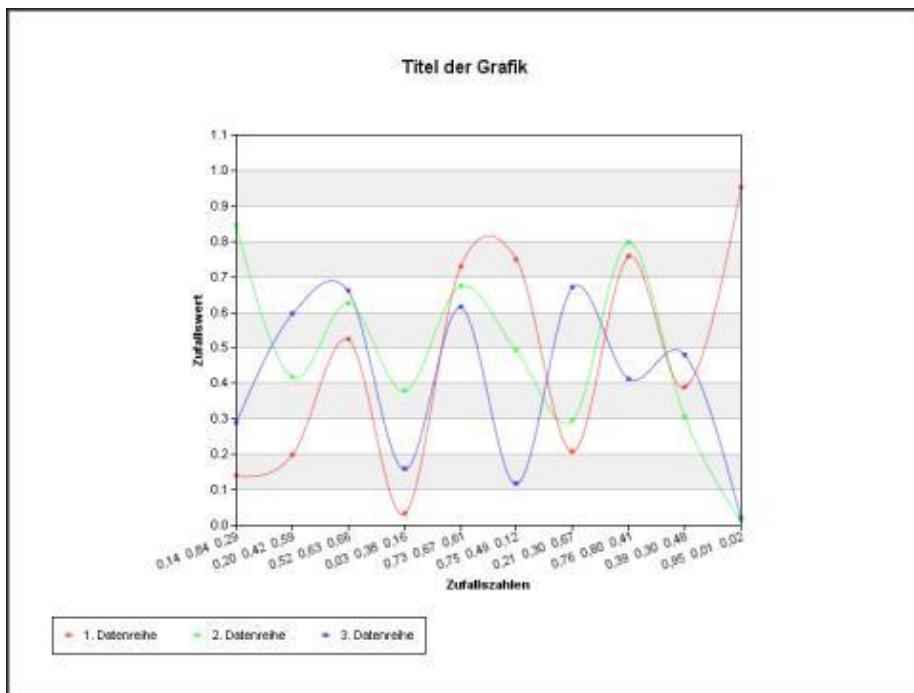
**Gestreute Darstellung.**

## Kontakt



- **\_ChartXYStyleDataSpline**

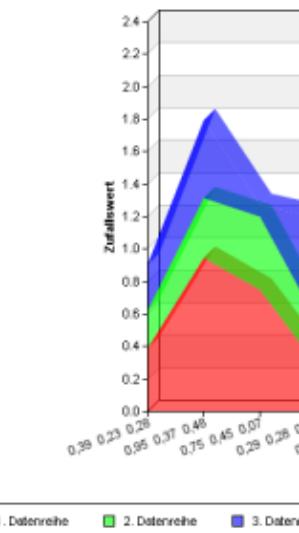
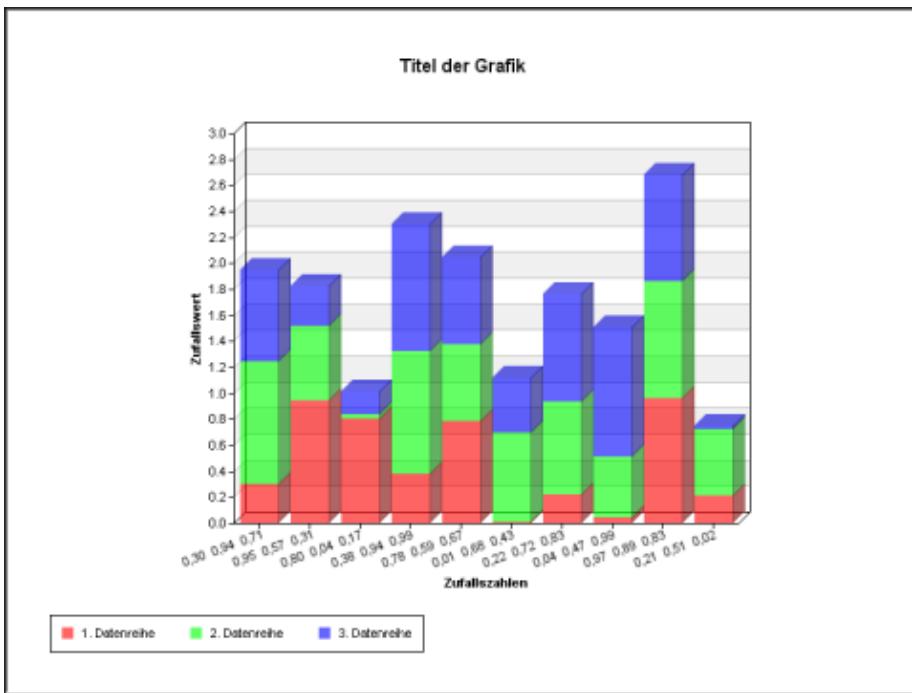
Darstellung als Spline-Linie.



- **\_ChartXYStyleDataStack**

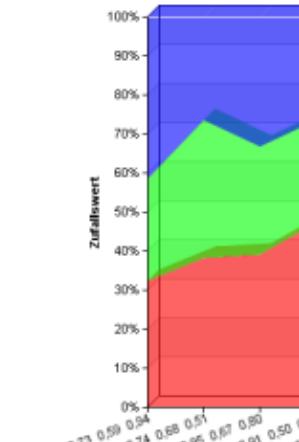
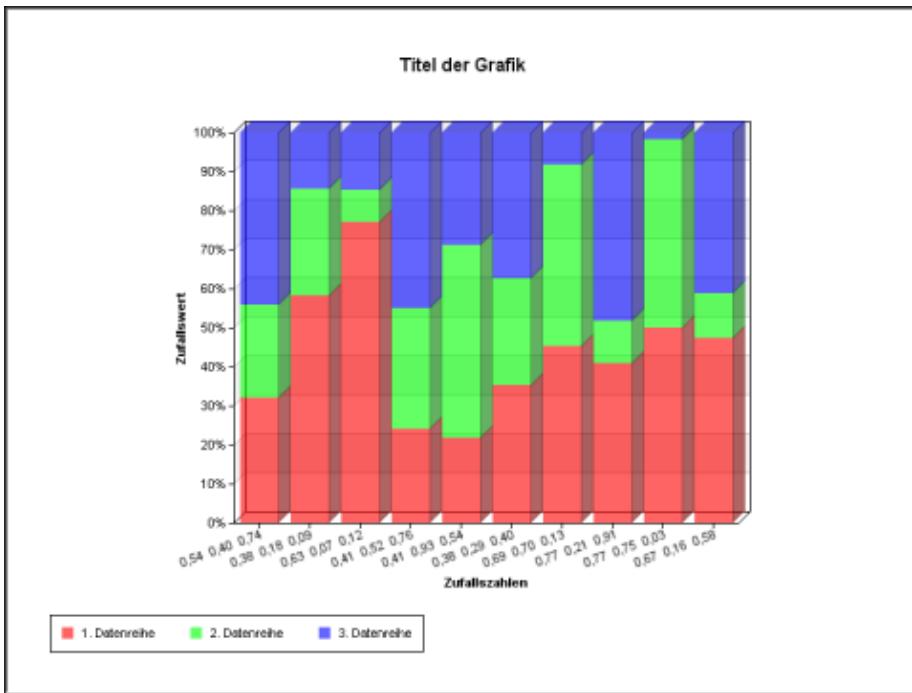
Gestapelte Darstellung.

## Kontakt



- \_ChartXYStyleDataPercent

**Prozentuale Darstellung.**



Die Stile \_ChartXYStyleDataStack und \_ChartXYStyleDataPercent sind nur in Kombination mit \_ChartXYStyleDataBar bzw. \_ChartXYStyleDataArea möglich.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Die Darstellung kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

**ChartXYStyleLabel**  Beschriftungsart der Daten eines Koordinatendiagramms Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Beschriftungsart der Daten in einem Koordinaten-Diagramm gesetzt und gelesen werden. Folgende Beschriftungsarten stehen zur Verfügung:

- \_ChartXYStyleLabelDefault (0) Ohne

Beschriftung.

- \_ChartXYStyleLabelData (1)

Wert der aktuellen Datenmenge der jeweiligen Position als Beschriftung anzeigen.

- \_ChartXYStyleLabelSum (2)

Summe der Werte aller Datenmengen der jeweiligen Position als Beschriftung anzeigen. Die Summe wird nur gebildet, wenn in der Eigenschaft ChartXYStyleData entweder ChartXYStyleDataStack oder ChartXYStyleDataPercent enthalten ist. Ist dies nicht der Fall, wird der Wert in der jeweiligen Datenreihe angezeigt.

- \_ChartXYStyleLabelDataExtra (4)

Zusätzliche Werte als Beschriftung der Daten anzeigen. Die Daten müssen mit der Option ChartDataExtra bei ChartDataAdd() hinzugefügt werden.

Die Konstanten können miteinander kombiniert werden.

### ChartXYSwapXY

X- und Y-Achse vertauschen

Typ [logic](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft können bei Koordinaten-Diagrammen die x- und y-Achse vertauscht werden. Standardmäßig werden die Achsen nicht vertauscht.



Wird für eine Datenreihe eine zusätzliche Achsenbeschriftung eingefügt

([ChartXYAxisY](#) != [ChartXYAxisDefault](#)) wird die Datenreihe unter Umständen falsch skaliert. Dies liegt daran, dass die Koordinatendiagramme nur eine X-Achse haben können.

## Kontakt

**ChartXYTitleAlignY**  Position des Titels der vertikalen Hauptachse Typ int

Siehe [Liste](#), [Chart](#), [ChartXYAxisY](#), [ChartXYTitleY](#)

Die Eigenschaft definiert Position und Richtung des Titels (Eigenschaft [ChartXYTitleY](#)) der vertikalen Hauptachse. Die Eigenschaft wirkt sich bei der Ausprägung [ChartXYAxisDefault](#) von [ChartXYAxisY](#) aus. Folgende Werte sind erlaubt:

- `_ChartAlignBottomLeft` (1)

Der Text befindet sich links unter der Achse und wird vertikal dargestellt.

- `_ChartAlignBottomCenter` (2)

Der Text befindet sich links unter der Achse und wird vertikal dargestellt.

- `_ChartAlignBottomRight` (3)

Der Text befindet sich rechts unter der Achse und wird vertikal dargestellt.

- `_ChartAlignLeft` (4)

Der Titel befindet sich links von der Achse, vertikal zentriert und wird vertikal dargestellt.

- `_ChartAlignCenter` (5)

Der Titel befindet sich auf der Achse, vertikal zentriert und wird vertikal dargestellt.

- `_ChartAlignRight` (6)

Der Titel befindet sich rechts von der Achse, vertikal zentriert und wird vertikal dargestellt.

- `_ChartAlignTopLeft` (7)

Der Titel befindet sich links über der Achse und wird vertikal dargestellt.

- `_ChartAlignTopCenter` (8)

Der Titel befindet sich horizontal zentriert über der Achse.

- `_ChartAlignTopRight` (9)

Der Titel befindet sich rechts über der Achse und wird vertikal dargestellt.

- `_ChartAlignTopLeft2` (10)

Der Text befindet sich links über der Achse.

- `_ChartAlignTopRight2` (11)

Der Text befindet sich rechts über der Achse.

- `_ChartAlignBottomLeft2` (12)

Der Text befindet sich links unter der Achse.

- `_ChartAlignBottomRight2` (13)

Der Text befindet sich rechts unter der Achse.

**ChartXYTitleFontX**

Schrift der x-Achsen-Beschriftung

Typ font

Siehe Liste, Chart

Mit dieser Eigenschaft kann die Schriftart, mit der der Titel der x-Achse gezeichnet wird, gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet. Die Schrift für die einzelnen Bezeichner wird in der Eigenschaft ChartXYLabelFontX gesetzt.

### **ChartXYTitleFontY**

Schrift der y-Achsen-Beschriftung

Typ [font](#)

Siehe [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Schrift, mit der der Titel der y-Achse gezeichnet wird, gesetzt und ermittelt werden. Als Standard wird die Schriftart Arial verwendet. Die Schrift für die Einteilung der Achsenabschnitte wird in der Eigenschaft [ChartXYLabelFontY](#) gesetzt.

## Kontakt

**ChartXYTitleX**



**Beschriftung der horizontalen Achse**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste](#), [Chart](#)

**Mit dieser Eigenschaft kann die Beschriftung der horizontalen Achse eines Koordinaten-Diagramms gesetzt und gelesen werden. Standardmäßig ist diese Eigenschaft nicht gesetzt.**

**ChartXYTitleY**

**Beschriftung der vertikalen Hauptachse**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste](#), [Chart](#)

Mit dieser Eigenschaft kann die Beschriftung der vertikalen Hauptachse eines Koordinaten-Diagramms gesetzt oder gelesen werden. Standardmäßig ist die Eigenschaft nicht gesetzt. Die Eigenschaft wirkt sich bei der Ausprägung [ChartXYAxisDefault](#) von [ChartXYAxisY](#) aus.

Über die Eigenschaft [ChartXYTitleAlignY](#) kann die Position der Hauptachse verändert werden.

## Kontakt

ChartXYTrendDegree

Grad des Polynoms

Typ int

Siehe Liste, Chart

Mit dieser Eigenschaft kann der Grad des Polynoms (ChartXYTrendType = ChartXYTrendTypePoly) in einem Koordinaten-Diagramm gesetzt und gelesen werden.

Standardmäßig beträgt der Grad des Polynoms 0.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Der Grad des Polynoms kann somit für jede Datenreihe separat gesetzt werden.

**ChartXYTrendType** 

**Typ der Trendlinie**

**Typ int**

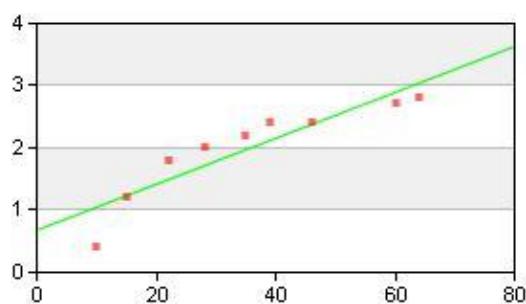
**Siehe Liste, Chart**

Mit dieser Eigenschaft kann der Typ der Trendlinie in einem Koordinaten-Diagramm gesetzt und gelesen werden. Für die Trendlinie stehen folgende Konstanten zur Verfügung:

- **\_ChartXYTrendTypeDefault / \_ChartXYTrendTypeNone** Keine Trendlinie.

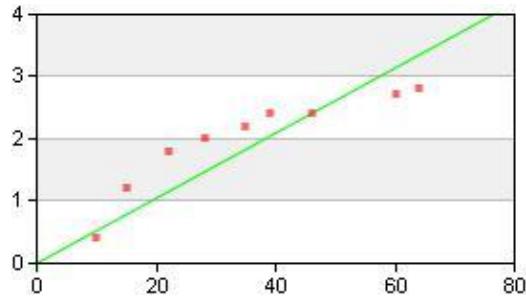
- **\_ChartXYTrendTypeLinear**

Lineare Trendlinie.



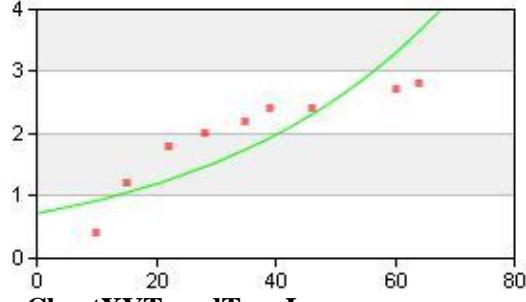
- **\_ChartXYTrendTypeLinearConst**

Gezwungene Lineare Trendlinie.



- **\_ChartXYTrendTypeExp**

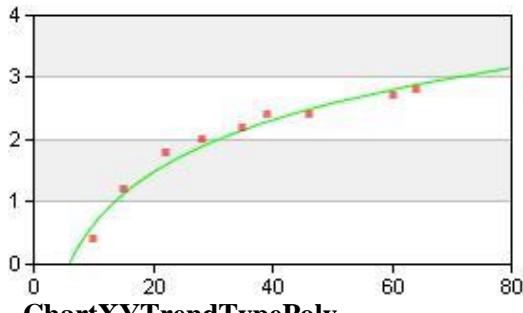
Exponentielle Trendlinie.



- **\_ChartXYTrendTypeLog**

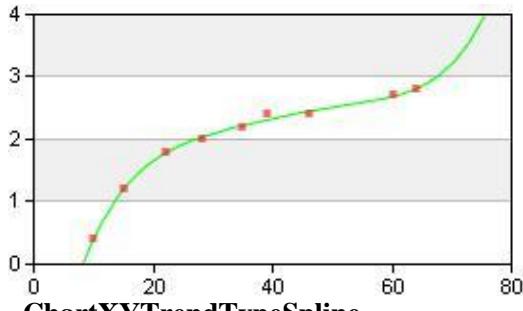
Logarithmische Trendlinie.

## Kontakt



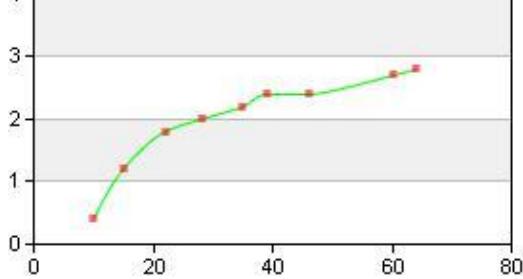
- \_ChartXYTrendTypePoly

Polynome Trendlinie. Der Grad des Polynoms wird über die Eigenschaft ChartXYTrendDegree definiert.



- \_ChartXYTrendTypeSpline

Spline Trendlinie.



Die Konstanten können nicht miteinander kombiniert werden.

Die Eigenschaft wird beim Schließen des ChartData-Objekts in das Chart-Objekt übertragen. Der Typ der Trendlinie kann somit für jede Datenreihe separat gesetzt werden.

## Kontakt

**ChildCount**  $\sqcup$  Anzahl der untergeordneten Knoten bei CteNode-  
Objekten Typ int

Siehe Liste, Objekte, AttribCount

Mit dieser Eigenschaft kann die Anzahl der untergeordneten Knoten eines CteNode-  
Objektes ermittelt werden.

Beispiel:

```
// Anzahl der Kindelemente ermittelntCount # tNode->spChildCount;
```

## Kontakt

---

CodepageOS

Codepage des Betriebssystems

Typ int

Siehe Liste, Objekte,

Bei dem Objekt System kann über diese Eigenschaft die Codepage des Betriebssystems ermittelt werden.

## Kontakt

**Compression**

**Kompressionsstufe**

**Typ** int

**Siehe** Liste, Objekte,

Diese Eigenschaft enthält die Kompressionsstufe des Objektinhalts. Die Eigenschaft kann nur gelesen werden.

Als Kompressionsstufen stehen Werte von 0 (keine Kompression) bis 4 (höchste Kompression) zur Verfügung.

## Kontakt

ContentLength



Länge des Nachrichtenkörpers

Typ int

Siehe Liste, Objekte,

Diese Eigenschaft enthält die Länge des HTTP-Bodys in Bytes, sofern ein entsprechender Header-Eintrag (Content-Length) beim Empfang einer Anfrage oder Antwort enthalten ist. Folgende Werte sind möglich:

- > 0 Wert entspricht der Länge des HTTP-Bodys
- 0 Es existiert kein HTTP-Body
- 1 Keine Angabe zur Länge des HTTP-Bodys enthalten. Es könnten Daten vorhanden sein.
- 2 Die Übertragung des Inhalts erfolgt in mehreren Teilen (Header-Eintrag Transfer-Encoding: chunked). Es sind Daten vorhanden. Die genaue Länge ist unbekannt.

Wenn Daten vorhanden sind, sollten diese mit HttpGetData() abgeholt werden. Die Eigenschaft kann nur gelesen werden.

## Kontakt

### CpnSetupFlagsCln



Installierte Client-Komponenten

Typ int

Liste, Objekte,  
CpnSetupFlagsSrv,

Siehe CpnSetupFlagsDoc,  
CpnSetupFlagsCom,  
SysPropGet()

Über diese Eigenschaft können die installierten Client-Komponenten ermittelt werden. Die Eigenschaft kann nur gelesen werden und liefert eine Kombination aus folgenden Konstanten zurück:

<b>0</b>	keine Client-Komponenten installiert
<u>_CpnClnStd</u>	Standard-Client ist installiert
<u>_CpnClnApi</u>	Service-Client ist installiert
<u>_CpnClnWeb</u>	Web-Schnittstelle ist installiert
<u>_CpnClnPhp</u>	PHP-Schnittstelle ist installiert
<u>_CpnClnPrtDrv</u>	Druckertreiber ist installiert
<u>_CpnClnPrtPpc</u>	Druckprozessor ist installiert
<u>_CpnClnOdbDrv</u>	ODBC-Treiber (vor 5.6) ist installiert
<u>_CpnClnOdbCln</u>	ODBC-Client (vor 5.6) ist installiert
<u>_CpnClnOdbSrv</u>	ODBC-Server (vor 5.6) ist installiert
<u>_CpnClnOdbDrv32</u>	ODBC-Treiber (32-bit) ist installiert
<u>_CpnClnOdbDrv64</u>	ODBC-Treiber (64-bit) ist installiert
<u>_CpnClnSOA</u>	SOA-Service ist installiert
<u>_CpnClnDrive</u>	Laufwerkstreiber ist installiert

Die Eigenschaft steht in Prozeduren, die vom Server, der PHP-Schnittstelle oder der Web-Schnittstelle verarbeitet werden, nicht zur Verfügung. Wird die Eigenschaft abgefragt, kommt es zum Laufzeitfehler ErrValueInvalid.

Beispiel:

```
if (_Sys->spCpnSetupFlagsCln & _CpnClnSOA > 0){ // SOA-Service installiert ...}
```

## Kontakt

CpnSetupFlagsCom



Installierte gemeinsame Komponenten

Typ int

Liste, Objekte, CpnSetupFlagsSrv,

Siehe CpnSetupFlagsDoc,

CpnSetupFlagsCln, SysPropGet()

Über diese Eigenschaft können die installierten gemeinsamen Komponenten ermittelt werden. Die Eigenschaft kann nur gelesen werden und liefert eine Kombination aus folgenden Konstanten zurück:

0                       keine gemeinsamen Komponenten installiert

\_CpnComCtxDocEdit Gemeinsame Komponenten des CtxDocEdit-Objektes installiert Die Eigenschaft steht in Prozeduren, die vom Server, der PHP-Schnittstelle oder der Web-Schnittstelle verarbeitet werden, nicht zur Verfügung. Wird die Eigenschaft abgefragt, kommt es zum Laufzeitfehler ErrValueInvalid.

Beispiel:

```
if (_Sys->spCpnSetupFlagsCom & _CpnComCtxDocEdit > 0){ // Komponenten des CtxDocEdit-Objektes in
```

## Kontakt

CpnSetupFlagsDoc



Installierte Komponenten Dokumente

Typ int

Liste, Objekte,

Siehe CpnSetupFlagsSrv,

CpnSetupFlagsCln,

CpnSetupFlagsCln, SysPropGet()

Über diese Eigenschaft können die installierten Dokumente ermittelt werden. Die Eigenschaft kann nur gelesen werden:

0 keine Dokumente installiert

\_CpnDocStd CONZEPT 16-Dokumentation ist installiert

Die Eigenschaft steht in Prozeduren, die vom Server, der PHP-Schnittstelle oder der Web-Schnittstelle verarbeitet werden, nicht zur Verfügung. Wird die Eigenschaft abgefragt, kommt es zum Laufzeitfehler ErrValueInvalid.

Beispiel:

```
if (_Sys->spCpnSetupFlagsDoc & _CpnDocStd > 0){ // Dokumentation installiert ...}
```

## Kontakt

CpnSetupFlagsSrv



Installierte Server-Komponenten

Typ int

Liste, Objekte,  
CpnSetupFlagsCln,  
Siehe CpnSetupFlagsDoc,  
CpnSetupFlagsCln,  
SysPropGet()

Über diese Eigenschaft können die installierten Server-Komponenten ermittelt werden. Die Eigenschaft kann nur gelesen werden und liefert eine Kombination aus folgenden Konstanten zurück:

0 keine Server-Komponenten installiert

\_CpnSrvWin Datenbank-Server für Windows ist installiert

\_CpnSrvLnx Datenbank-Server für Linux ist installiert \_CpnSrvCtl

Control-Center ist installiert

Die Eigenschaft steht in Prozeduren, die vom Server, der PHP-Schnittstelle oder der Web-Schnittstelle verarbeitet werden, nicht zur Verfügung. Wird die Eigenschaft abgefragt, kommt es zum Laufzeitfehler ErrValueInvalid.

Beispiel:

```
if (_Sys->spCpnSetupFlagsSrv & _CpnSrvWin > 0){ // Windows-Server installiert ...}
```

## Kontakt

---

### Created

Erstellungszeitpunkt des Objekts

Typ caltime

Siehe Liste, Objekte, CreatedUser,

Diese Eigenschaft enthält Datum und Uhrzeit des Anlegens des Objekts. Die Eigenschaft kann nur gelesen werden.

## Kontakt

**CreatedUser**



**Benutzer, der das Objekt erstellt hat.**

**Typ** alpha(20)

**Siehe** Liste, Objekte, Created,  
SysPropGet()

Diese Eigenschaft enthält den Namen des Benutzers, der das Objekt angelegt hat. Die Eigenschaft kann nur gelesen werden.

## Kontakt

CteNodeSepAttrib  Attribut-Separator bei  
CteNode-Objekten Typ alpha(1)

Siehe Liste, Objekte

Bei dem Objekt System kann über diese Eigenschaft der Attribut-Separator für die Suche von CteNode-Objekten mit CteRead() und der Option CteNodePath oder CteNodePathCI gesetzt werden. Standardmäßig hat die Eigenschaft den Wert '.'.

## Kontakt

---

### CteNodeSepPath

Pfad-Separator bei CteNode-Objekten

Typ [alpha\(1\)](#)

Siehe [Liste, Objekte](#)

Bei dem Objekt [System](#) kann über diese Eigenschaft der Pfad-Separator für die Suche von [CteNode](#)-Objekten mit [CteRead\(\)](#) und der Option [\\_CteNodePath](#) oder [\\_CteNodePathCI](#) gesetzt werden. Standardmäßig hat die Eigenschaft den Wert '/'.

## Kontakt

### Custom (Systemeigenschaft)

#### Benutzerdefinierte Eigenschaft

Typ alpha(255) oder  
alpha(65520)

Siehe Liste, Objekte,  
SysPropGet(), SysPropSet()

Diese Eigenschaft dient zum Ablegen benutzerdefinierter Daten. Der Wert kann bei Binären Objekten maximal 255 und bei allen anderen Objekten maximal 65.520 Stellen lang sein. Beim Setzen kann sie aus mehreren Zeichenketten zusammengesetzt oder beispielsweise aus einem Memory-Objekt gelesen werden:

```
// Inhalt eines Memory-Objektes in Custom schreibt CteItem->spCustom # tMem->MemReadStr(1, Min(t
```

## Kontakt

---

**DisplayRaisingDelay**  Verzögerung bei der Anzeige des  
Selektions-Dialogs Typ int

Siehe [Liste](#), [SysPropGet\(\)](#), [SysPropSet\(\)](#)

Diese Eigenschaft kann bei einem Selektionspuffer (siehe [SelOpen\(\)](#)) gesetzt werden. Wird die Selektion mit [SelRun\( SelDisplayDelayed, ...\)](#) durchgeführt, erscheint die Anzeige der Statusinformationen um die in dieser Eigenschaft gesetzten Anzahl von Millisekunden verzögert. Ist die Selektion zu diesem Zeitpunkt bereits durchgeführt, erscheint keine Statusanzeige.

Standardmäßig ist die Eigenschaft auf 2000 Millisekunden gesetzt.

## Kontakt

ErrCode



Fehlerwert

Typ int

Liste,

Siehe Objekte,

SysPropGet()

Diese Eigenschaft hat unterschiedliche Bedeutungen in Abhängigkeit von ihrem Objekt:

- System

Die Eigenschaft kann in einer Prozedur abgefragt werden, die bei einem Laufzeitfehler (ErrCall()) aufgerufen wird. Die Eigenschaft wird ebenfalls durch die Anweisung ProcCompile() gesetzt. In dieser Eigenschaft steht der Fehlerwert. Die Fehlermeldung steht in der Eigenschaft ErrText. Aus dem Fehlerwert kann mit der Anweisung ErrMapText() ein Fehlertext in unterschiedlichen Sprachen erzeugt werden.

- Selection

Wird die Eigenschaft von einem Selection-Objekt abgerufen, wird der Fehlerwert zurückgegeben. Die Fehlermeldung steht in der Eigenschaft ErrText.

Beispiel:

```
tCode # _Sys->spErrCode;tCode # tHdlSel->spErrCode;
```

### ErrLine

Programmzeile des Laufzeitfehlers

Typ int

Siehe Liste, Objekte, SysPropGet()

Hier kann die Programmzeile ermittelt werden, in der ein Fehler aufgetreten ist.

Die Eigenschaft kann in einer Prozedur abgefragt werden, die bei einem Laufzeitfehler (ErrCall()) aufgerufen wird. Die Eigenschaft wird ebenfalls durch die Anweisung ProcCompile() gesetzt.

Beispiel:

```
tLine # _Sys->spErrLine;
```

## Kontakt

**ErrPos (Systemeigenschaft)**  —

**Position des Fehlers**

Typ int

Siehe Liste, Objekte,

SysPropGet()

Mit dieser Eigenschaft kann die Position eines Fehlers innerhalb eines Abfragekriteriums ermittelt werden. Die Eigenschaft kann aus dem entsprechenden Selection-Objekt ermittelt werden.

**Beispiel**

```
tStr # StrCut(aHdl->spErrSource, 1, aHdl->spErrPos - 1) +      '[' + aHdl->spErrText + ']' +
```

## Kontakt

ErrProc



Prozedurfunktion, in der der Fehler aufgetreten ist.

Typ alpha(61)

Siehe Liste, Objekte, ErrSource, SysPropGet()

In dieser Eigenschaft kann die Prozedurfunktion, in der ein Fehler aufgetreten ist, ermittelt werden.

Die Eigenschaft kann in einer Prozedur abgefragt werden, die bei einem Laufzeitfehler (ErrCall()) aufgerufen wird. Die Eigenschaft wird ebenfalls durch die Anweisung ProcCompile() gesetzt. Die Funktion wird in der Form <Prozedurname>:<Funktionsname> angegeben.

Beispiel:

```
tProcName # _Sys->spErrProc;
```

## Kontakt

ErrSource



Quellcode

Typ alpha(20)

Liste,

Siehe Objekte,

ErrProc,

SysPropGet()

Diese Eigenschaft hat unterschiedliche Bedeutungen in Abhängigkeit von ihrem Objekt:

- System

In der Eigenschaft ErrProc wird der Name der Prozedur zurückgegeben, in der ein Laufzeitfehler aufgetreten ist. Die Eigenschaft wird ebenfalls durch die Anweisung ProcCompile() gesetzt. Sofern die Programmzeile des Fehlers aus einer mit Include eingebundenen Prozedur stammt, ist dies der Name der Prozedur.

Die Eigenschaft kann in einer Prozedur abgefragt werden, die bei einem Laufzeitfehler (ErrCall()) aufgerufen wird.

- Selection

Wird die Eigenschaft von einem Selection-Objekt abgerufen, wird das Abfragekriterium der Abfrage zurückgegeben.

Beispiel:

```
tProcnameInclude # _Sys->spErrSource;// Selektionskriterium an Fehlerstelle auftrennenStr # StrC
```

## Kontakt

**ErrSourceLine**

**Quellcodezeile**

**Typ** alpha(8192)

Liste,

Objekte,

**Siehe** ErrProc,

ProcCompile(),

SysPropGet()

Konnte eine Prozedur aufgrund eines Fehlers im Prozedurtext nicht mit der Anweisung ProcCompile() übersetzt werden, steht in dieser Eigenschaft die gesamte fehlerhafte Programmzeile zur Verfügung. Die Art des Fehlers und die Position innerhalb der Zeile kann in den Eigenschaften ErrCode und ErrLine / ErrPos() ermittelt werden.

Ist der Fehler innerhalb eines Defines aufgetreten, wird das Define entsprechend aufgelöst.

## Kontakt

ErrText

Fehlertext

Typ alpha(128)

Liste,

Siehe Objekte,

SysPropGet()

Diese Eigenschaft hat unterschiedliche Bedeutungen in Abhangigkeit von ihrem Objekt:

- System

Die Eigenschaft kann in einer Prozedur abgefragt werden, die bei einem Laufzeitfehler (ErrCall()) aufgerufen wird. Die Eigenschaft wird ebenfalls durch die Anweisung ProcCompile() gesetzt. In dieser Eigenschaft steht der Fehlertext. Der Fehlerwert steht in der Eigenschaft ErrCode. Aus dem Fehlerwert kann mit der Anweisung ErrMapText() der dazugehorende Fehlertext in unterschiedlichen Sprachen erzeugt werden.

- Selection

Wird die Eigenschaft von einem Selection-Objekt abgerufen, nach dem es mit SelStore() gespeichert wurde, wird der Fehlertext zuruckgegeben. Der Fehlerwert steht in der Eigenschaft ErrCode.

Beispiel:

```
tErrorMessage # _Sys->spErrText;tErrorMessage # tHdlSel->spErrText;
```

## Kontakt

### Eigenschaften von System-Objekten - F bis L

In dieser Liste sind alle Eigenschaften von F bis L von System-Objekten aufgeführt. [Liste sortiert](#)

[nach Gruppen,](#)

Siehe [Alphabetische](#)  
[Liste aller](#)  
[Eigenschaften](#)

- [FileFilter](#)
- [FileFilterNum](#)
- [FileNameExt](#)
- [Flags](#)
- [FsiError](#)
- [FullName](#)
- [Hash](#)
- [HdlCount](#)
- [HostName](#)
- [HttpHeader](#)
- [HttpParameters](#)
- [ID](#)
- [JobData](#)
- [JobErrorCode](#)
- [JobID](#)
- [JobMsxReadQ](#)
- [JobMsxWriteQ](#)
- [JobProcesses](#)
- [JobProcExtended](#)
- [JobSckHandle](#)
- [JobStatus](#)
- [JobThreads](#)
- [LclCurrDecimal](#)
- [LclCurrFract](#)
- [LclCurrFractIntl](#)
- [LclCurrGroup](#)
- [LclCurrNegSignPos](#)
- [LclCurrNegSpaceSep](#)
- [LclCurrNegSymPrec](#)
- [LclCurrPosSignPos](#)
- [LclCurrPosSpaceSep](#)
- [LclCurrPosSymPrec](#)
- [LclCurrSymbol](#)
- [LclCurrSymbolIntl](#)
- [LclCurrTSep](#)
- [LclDateDayLZero](#)
- [LclDateDayN](#)
- [LclDateDayNS](#)
- [LclDateLFormat](#)
- [LclDateLOrder](#)
- [LclDateMonthLZero](#)
- [LclDateMonthN](#)
- [LclDateMonthNS](#)

- [LclDateSCentury](#)
- [LclDateSep](#)
- [LclDateSFormat](#)
- [LclDateSOrder](#)
- [LclNumDecimal](#)
- [LclNumFract](#)
- [LclNumGroup](#)
- [LclNumLZero](#)
- [LclNumNegMode](#)
- [LclNumNegSign](#)
- [LclNumPosSign](#)
- [LclNumTSep](#)
- [LclTimeHourMode](#)
- [LclTimeLFormat](#)
- [LclTimeLZero](#)
- [LclTimeMarkPos](#)
- [LclTimeSep](#)
- [LclTimeSepAM](#)
- [LclTimeSepPM](#)
- [LclTimeSFormat](#)
- [Len](#)
- [LogicalProcessors](#)

## Kontakt

FileFilter

Dateifilter

Typ alpha(8192)  
Liste, Objekte,  
Siehe WinPropGet(),  
WinPropSet()

Über diese Eigenschaft wird der Dateifilter definiert.

Beispiel:

```
tHdl->wpFileFilter # 'Resourcen (*.rsc)|*.rsc|Texte (*.txt)|*.txt';
```

In diesem Beispiel wird ein Common-Dialog definiert, der einen Filter auf Dateien vom Typ \*.rsc und \*.txt setzt. In der Combobox Dateityp des Dialogs werden die Informationen Resourcen (\*.rsc) und Texte (\*.txt) angezeigt. Es können auch mehrere Dateierweiterungen in einem Eintrag aufgenommen werden:

```
tHdl->wpFileFilter # 'Bild-Dateien (*.bmp;*.gif;*.jpg;*.tif');
```

### FileFilterNum

Index des Dateifilters

Typ int

Liste, Objekte,

Siehe WinPropGet(),

WinPropSet()

Über diese Eigenschaft wird festgelegt welcher der gesetzten Dateifilter als Standard vorgegeben wird.

**Beispiel:**

```
tHdl->wpFileFilter # 'Resourcen (*.rsc)|*.rsc|Texte (*.txt)|*.txt'; tHdl->wpFileFilterNum # 2;
```

Standardmäßig wird der zuerst definierte Filter als Vorgabe verwendet. In dem obigen Beispiel wäre das \*.rsc. Über die Anweisung tHdl->wpFileFilterNum # 2 wird der Filter, welcher an zweiter Stelle definiert wurde, als Vorgabe verwendet. In diesem Fall \*.txt.

## Kontakt

**FileNameExt**



**Standardmäßige Dateiendung.**

**Typ** alpha(8192)

Liste, Objekte,

**Siehe** WinPropGet(),

WinPropSet()

Über diese Eigenschaft wird eine standardmäßige Dateiendung festgelegt. Wird beim Öffnen oder Speichern einer Datei keine Erweiterung angegeben, wird die in FileNameExt definierte Erweiterung verwendet.

### Flags

#### Optionen für Objekte

Typ int

Siehe Liste, Objekte

Über diese Eigenschaft werden Optionen für unterschiedliche Objekte festgelegt. Im Folgenden sind die Optionen zu den Objekten aufgeführt:

- CteNode-Objekt
- Systemdialoge
- WinComPrint-Dialog
- WinComFont-Dialog
- Notebook-Objekt
- Application-Objekt
- Validierungselement
- CteNode-Objekt

In dieser Eigenschaft werden die vorhandenen Listen für untergeordnete Knoten oder Attributknoten definiert. Der Inhalt der Eigenschaft besteht aus einer Kombination folgender Konstanten:

<u>CteChildList</u>	Verkettete Liste der untergeordneter Knoten
<u>CteChildTree</u>	Sortierte Liste der untergeordneter Knoten
<u>CteChildTreeCI</u>	Sortierte Liste der untergeordneten Knoten (ohne Unterscheidung der Groß-/Kleinschreibung)
<u>CteAttribList</u>	Verkettete Liste der Attributknoten
<u>CteAttribTree</u>	Sortierte Liste der Attributknoten
<u>CteAttribTreeCI</u>	Sortierte Liste der Attributknoten (ohne Unterscheidung) der Groß-/Kleinschreibung)

Die Eigenschaft wird bereits beim Erzeugen eines Knoten-Objekts mit der Anweisung CteOpen( \_CteNode, ...) gesetzt, kann aber auch zu einem späteren Zeitpunkt gesetzt werden. Bestehende Listen können entfernt werden, wird eine neue Liste hinzugefügt, werden eventuell bestehende Listen des Knotens geleert.

- Systemdialoge

Folgende Ausprägungen können bei den Systemdialogen WinComFileOpen, WinComFileSave und WinComPath gesetzt werden:

<u>WinComCreatePrompt</u>	Abfrage Datei anlegen
<u>WinComOverwritePrompt</u>	Abfrage Datei überschreiben
<u>WinComFileMustExist</u>	Datei muss vorhanden sein
<u>WinComPathMustExist</u>	Pfad muss vorhanden sein
<u>WinComShareAware</u>	Dateiauswahl von geöffneten Dateien möglich
<u>WinComExplorer</u>	Windows Explorer Darstellung
<u>WinComNoChangeDir</u>	letztes Verzeichnis nicht speichern
<u>WinComNoNetworkBtn</u>	kein Netzwerk-Button

## Kontakt

### WinComAllowMultiSelect Mehrfachauswahl

WinComEditBox Edit-Objekt für Verzeichnis

WinComSizeableDialog Größe veränderbar

WinComUserHint kurze Bedienungsanweisung

WinComNoNewFolderBtn keine neuen Verzeichnisse angelegen

Bei dem Dialog WinComPath werden je nach Betriebssystem nicht alle Optionen bzw. nicht alle Kombinationen von Optionen unterstützt:

Windows      WinComNoNewFolderBtn funktioniert nur in Verbindung mit  
XP / Vista     WinComSizeableDialog. WinComUserHint funktioniert nur, wenn  
                  WinComEditBox nicht gesetzt ist. Die restlichen Optionen werden  
                  ohne Einschränkung unterstützt.

- WinComPrint

Der Dialog unterstützt folgende Optionen:

WinComPrintPageAll alle Seiten drucken

WinComPrintPageRange Bereich von Seiten drucken

WinComPrintPageCur aktuelle Seite drucken

WinComPrintDisablePageRange Seitenbereich deaktivieren

WinComPrintDisablePageCur Markierung deaktivieren

- WinComFont

Der Dialog unterstützt folgende Optionen:

WinComFontFixedPitchOnly nur Schriftarten mit gleicher Zeichenbreite

WinComFontNoVectorFonts keine Vektor-Schriftarten

WinComFontNoVerticalFonts keine vertikal orientierten Schriftarten

WinComFontScalableOnly nur skalierbare Schriftarten

WinComFontNoSimulations nur Stile (fett, kursiv, ...) anzeigen

WinComFontTrueTypeOnly nur TrueType- und OpenType-Schriftarten

WinComFontEffects Anzeige von Schriftart-Effekten

WinComFontNoFaceNameInit keine Schriftart vorbelegt

WinComFontNoSizeInit keine Schriftgröße vorbelegt

WinComFontNoStyleInit kein Schriftstil vorbelegt

- Notebook-Objekt

Die Eigenschaft kann beim Notebook-Objekt nur prozedural gesetzt werden.

WinFlagNbFocusFirst erstes untergeordnetes Objekt fokussieren

WinFlagNbCtrlTabOff kein Wechsel mit oder

- Application-Objekt

Das Objekt unterstützt folgende Ausprägungen:

## Kontakt

<u>WinAppTabNoScrollSelect</u>	Seite nicht selektieren
<u>WinAppEditTextCenterVert</u>	Text in Eingabeobjekten vertikal zentrieren
<u>WinAppEditC16CharsOnly</u>	nur Zeichen aus textbasierten Oberfläche
<u>WinAppEditSelectAll</u>	bei Mausklick gesamten Inhalt selektieren
<u>WinAppEditDateDefaultCurr</u>	aktueller Tag als Vorgabewert
<u>WinAppEditDateFormatDDMM</u>	Schnelleingabemodus für Datum
<u>WinAppEditDateFast</u>	Inhalt überschreiben
<u>WinAppEditTimeFast</u>	keine Korrektur bei ungültigen Zeitwerten
<u>WinAppRadioFocusChecked</u>	ausgewählten Radiobutton selektieren
<u>WinAppEditTextROContextMenu</u>	Kontextmenü bei ReadOnly-Objekten
<u>WinAppObj2FldOff</u>	aufgerufen
<u>WinAppFld2ObjOff</u>	Update zum Datenbankfeld verhindern
<u>WinAppBuf2FldOff</u>	Update zum Objekt verhindern
<u>WinAppFld2BufOff</u>	Update der Datensatzpuffer der Objekte zum Feldpuffer verhindern
<u>WinAppTBarFocusRestore</u>	Update der Feldpuffer zum Datensatzpuffer der Objekte verhindern
<u>WinAppPrtJobPreviewC16</u>	Fokus in Eingabeobjekt setzen
<u>WinGanttIvlBoxClick</u>	<u>Caption- und HelpTip-Eigenschaften</u> der Druckvorschau im CONZEPT 16-Zeichensatz
<u>WinAppWaitCursorEvt</u>	EvtMenuItem bei Box-Objekt auslösen
<u>WinAppWaitCursorEvtOs</u>	CONZEPT 16-Sanduhr während eines Ereignisses
<u>WinAppWaitCursorEvtArrow</u>	Betriebssystem-Sanduhr während eines Ereignisses
<u>WinAppSearchMdiFrame</u>	keine Sanduhr während eines Ereignisses
<u>WinAppRtfUrlHighlight</u>	Suche nach Objektnamen auf aktiven MdiFrame beschränken
<u>WinAppPrtUrlHighlight</u>	Hyperlinks im RtfEdit-Objekt nicht hervorheben
<u>WinAppCalcKeySupport</u>	Hyperlinks im PrtRtf-Objekt nicht hervorheben
<u>WinAppFontIgnoreScript</u>	Aufruf von EvtKeyItem bei Tasten des Nummernblocks.
<u>WinAppNotebookPageDelayed</u>	Skript-Einstellung von Fonts ignorieren.
<u>WinAppRecFocusTermResetOff</u>	Laden der Notizbuchseite bei der Anzeige.
<u>WinAppKeepFocusOnError</u>	Ignorieren der LstFlags-Option
<u>WinAppHighlightFocus</u>	WinLstRecFocusTermReset bei allen RecList-Objekten.
<u>WinAppGlobalVarEvent</u>	Datumsfeld bei ungültiger Eingabe nicht verlassen.
	Hervorhebung beim Erhalten des Fokus.
	Globale Variablenbereiche des alten MdiFrame im EvtFocusTerm aktivieren.

## Kontakt

### • Validierungselement

Die Eigenschaft Flags von Validierungselementen unterstützt folgende Ausprägungen:

<u>VldStateUndefined</u>	Validierungsstatus nicht definiert
<u>VldStateIrrelevant</u>	Referenzelement nicht für Validierung relevant
<u>VldStateNotVerified</u>	Referenzelement noch nicht validiert
<u>VldStateVerified</u>	Referenzelement validiert
<u>VldStateModified</u>	Referenzelement modifiziert
<u>VldStateDeleted</u>	Referenzelement gelöscht



Die Konstanten für Validierungselemente können nicht miteinander kombiniert werden.  
Bei der Verwendung der Validierungsbefehle können zudem eigene Werte angegeben werden, die vom Validierungs-Editor nicht interpretiert werden.

Die Flags können wie folgt überprüft und gesetzt werden:

```
// Flag auswerten: if (tSysDialog->wpFlags & _WinComCreatePrompt > 0) { ... } // Flag setzen _App->wp
```

## Kontakt

FsiError



Fehlerwert des letzten Zugriffs auf eine externe Datei Typ int

Siehe Liste, Objekte, SysPropGet()

Die Eigenschaft enthält den Betriebssystemfehler des letzten Zugriffs auf eine externe Datei. Der Fehler kann ausgelesen und mit der Anweisung ErrMapText() in seinen Fehlertext umgewandelt werden.

Neben den Dateibefehlen setzt auch DllLoad() die Eigenschaft FsiError.

Beispiel:

```
tFsiError # _Sys->spFsiError; // read error code ErrMsg # ErrMapText(tFsiErr
```

## Kontakt

**FullName**

**Pfad und Name des Objekts**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste, Objekte](#),  
[SysPropGet\(\)](#), [Name](#)

Dies ist der vollständige Name des Verzeichnisses oder Objektes inklusive Pfad. Die Eigenschaft kann nur gelesen werden.

## Kontakt

---

### Hash

**Hashwert des Validierungselementes**

Typ [alpha\(79\)](#)

Siehe [Liste, Objekte](#)

Diese Eigenschaft enthält den Hashwert des Validierungselementes. Sie wird vom Validierungs-Editor für Tabellenelemente neu gesetzt, wenn der Validierungsstatus ermittelt wird.

## Kontakt

**HdlCount**

**Anzahl der Handles**

Typ int  
Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft die Anzahl der verwendeten Handles ermittelt werden.

## Kontakt

---

### HostName

Hostname der HTTP-Anfrage

Typ alpha(127)

Liste, Objekte,

Siehe SysPropGet(),

SysPropSet()

Eigenschaft des HTTP-Objekts. In dieser Eigenschaft wird der Hostname der HTTP-Kommunikation abgelegt. Die Eigenschaft steht nur bei HTTP-Objekten vom Typ HttpSendRequest und HttpRecvRequest zur Verfügung. Beim Typ HttpRecvRequest kann die Eigenschaft nur gelesen werden.

Beim Typ HttpSendRequest muss ein Hostname definiert werden, wenn HTTP/1.1 verwendet wird.

Auch bei HTTP/1.0 sollte der Hostname in jedem Fall gesetzt werden.

## Kontakt

### HttpHeader

#### HTTP-Header-Einträge

Typ handle

Siehe Liste, Objekte,

Diese Eigenschaft enthält den Deskriptor einer Cte-Liste, die alle HTTP-Header-Einträge enthält. Bei den Typen HttpSendRequest und HttpSendResponse können beliebig viele Einträge in Form von CteItems in die Liste aufgenommen werden. Der Name des Header-Eintrags steht dabei in der Name-Eigenschaft des Elements (ohne Doppelpunkt), der Wert in der Custom-Eigenschaft.

Wenn eine Prozedur Header-Einträge definiert, haben diese Vorrang vor den Standardwerten des HTTP-Objekts. Wird beispielsweise ein Eintrag mit dem Namen User-Agent erzeugt, wird der standardmäßige Wert dafür nicht verwendet.

Die Eigenschaft selbst kann nur gelesen werden. Der Deskriptor der Liste kann anschließend verwendet werden, um neue Elemente in die Liste einzutragen (siehe CteInsertItem()).

#### Beispiel:

```
tHdList # tSendRequest->spHttpHeader;tHdList->CteInsertItem('Host', 0, tHostName);tHdList->Cte
```

### HttpParameters

Parameter einer HTTP-Kommunikation

Typ [handle](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Diese Eigenschaft enthält den Deskriptor einer [Cte-Liste](#), die alle HTTP-Parameter enthält. Bei den Typen [HttpSendRequest](#) und [HttpSendResponse](#) können beliebig viele Einträge in Form von Elementen in die Liste aufgenommen werden. Der Name des Parameters steht dabei in der [Name-Eigenschaft](#), der Wert in der [Custom-Eigenschaft](#) des Elements. Parameter können entweder Bestandteil der [URI](#) oder des HTTP-Bodys sein (siehe [HttpGetData\(\)](#) und [HttpClose\(\)](#)). Die Eigenschaft kann nur gelesen werden.

**Beispiel:**

```
tHttpParams # tRecvRequest->spHttpParameters;for    tParam # tHttpParameters->CteRead(_CteFirst);l
```

## Kontakt

ID (Systemeigenschaft)  —

**Identität des Objekts**

Typ [bigint](#)

[Liste, Objekte](#),

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

[SysPropSet\(\)](#)

Dies ist die Identität (ID) des Objekts. Die ID von Verzeichnissen und binären Objekten kann nur gelesen werden.

### JobData

---

Daten für den Job

Typ [alpha\(8192\)](#)

[Liste](#),

Siehe [Objekte](#),

[SysPropGet\(\)](#),

[SysPropSet\(\)](#)

In dieser Eigenschaft ist der bei [JobStart\(\)](#) übergebene Text enthalten. Der Text kann beispielsweise Informationen über die im Job zu verarbeitenden Daten enthalten.

## Kontakt

**JobErrorCode**

**Fehlerwert des Jobs**

Typ int

Siehe Liste, Objekte,

Diese Eigenschaft enthält den aktuellen Fehlerwert des Jobs. Bei einem Fehlerwert von 0 (ErrOk) ist der Job noch aktiv, bei ErrTerminated hat sich der Job ohne Fehler beendet. Bei allen anderen Codes ist ein Fehler entweder beim Start des Jobs oder während der Prozedurausführung aufgetreten. Bei Job-Prozessen existiert außerdem noch der Fehlerwert ErrKilled falls sich der Prozess nicht ordnungsgemäß beendet hat.

**Der Fehlerwert wird im Fehlerfall automatisch durch den Job gesetzt und kann nur gelesen werden.**

## Kontakt

### JobID

Identität des Objekts

Typ int

Siehe Liste, Objekte,

Dies ist die Identität des Job-Objekts. Sie wird vom System vergeben und kann nicht verändert werden.

Beim Sys-Objekt ist der Wert 0, wenn die Prozedur nicht mit JobStart() gestartet wurde.

## Kontakt

**JobProcExtended**



Ausführung von Oberflächen-Befehlen

Typ logic

Siehe Objekte, SysPropGet()

Über diese Eigenschaft des Job-, JobControl, bzw. Sys-Objektes kann ermittelt werden, die aktuell ausgeführte Prozedur (z. B. ein SOA-Task oder ein durch JobStart() ausgeführter Job) die Ausführung von Oberflächen-Befehlen unterstützt (true) oder nicht (false).

## Kontakt

Objekte von JobProcExtended

Folgende Objekte unterstützen die Eigenschaft JobProcExtended

JobProcExtended,

Liste der

Siehe Objekte, Liste

der

Eigenschaften

- Job
- JobControl
- System
- Task
- Sys

## Kontakt

---

### JobMsxReadQ

Anzahl der nicht gelesenen Nachrichten

Typ int  
Siehe Liste, Objekte, JobMsxWriteQ,

Dies ist die Anzahl von Messages, die sich in der MSX-Pipeline befinden und noch nicht vom Job gelesen wurden (siehe Verarbeitungshinweise).

## Kontakt

**JobMsxWriteQ**  Anzahl der nicht abgeholtene

Nachrichten Typ int

Siehe [Liste](#), [Objekte](#), [JobMsxReadQ](#),

Dies ist die Anzahl von Messages, die sich in der MSX-Pipeline befinden und noch nicht von einem Job-Kontroll-Objekt aus abgeholt wurden. Dieser Wert ist bei Jobs, die in eigenen Prozessen laufen, immer 0.

### JobProcesses

**Anzahl der aktiven Job-Prozesse**

Typ **int**

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Dies ist die Anzahl der aktuell aktiven Job-Prozesse, die von diesem [Task-Prozess](#) mit der Anweisung [JobStart\(\)](#) gestartet wurden. Bei Abfrage dieser Eigenschaft wird die Anzahl jedesmal neu ermittelt.



Es werden nur die Prozesse ermittelt, die aus dem Task oder Job erzeugt wurden. Prozesse, die von diesen Prozessen oder von anderen Tasks gestartet wurden, können nicht ermittelt werden.

Die Eigenschaft kann nur gelesen werden.

## Kontakt

**JobSckHandle**  Kopie des übergebenen Socket-  
Deskriptors Typ handle

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Dies ist eine Kopie des Socket-Deskriptors, der bei [JobStart\(\)](#) angegeben wurde. Mit diesem kann auf die gleiche Socket-Verbindung zugegriffen werden, wie in der Prozedur, die den [Job](#) gestartet hat.

Hierbei sollte jedoch eine Seite nur lesen und eine nur schreiben.



Die Socket-Verbindung muss nur auf der Seite geschlossen werden, die [JobStart\(\)](#) aufgerufen hat.

Wird der Socket auf der Seite des Jobs geschlossen, wird die Verbindung auf beiden Seiten getrennt. Jedoch bleibt der Socket-Deskriptor in der Prozedur, die den Job gestartet hat, erhalten. Dieser muss dann separat mit [SckClose\(\)](#) geschlossen werden. Am Ende des Jobs wird die erzeugte Kopie des Socket-Dekriptors entfernt. Die Socket-Verbindung bleibt am Ende des Jobs erhalten und kann in der aufrufenden Prozedur bis zum [SckClose\(\)](#) verwendet werden.

## Kontakt

JobStatus



Benutzerdefinierter Job-Status

Typ int

Liste, Objekte,

Siehe SysPropGet(),

SysPropSet()

Dies ist ein benutzerdefinierter Statuswert, mit dem der Job einen Verarbeitungszustand signalisieren kann. Der Wert kann sowohl vom Job als auch von einem mit JobOpen() erzeugten Kontroll-Objekt aus gelesen werden. Im Job kann er zusätzlich auch geändert werden.

### JobThreads

Anzahl der aktiven Job-Threads

Typ int

Siehe Liste, Objekte,

Dies ist die Anzahl der aktuell aktiven Job-Threads, die von diesem Task-Prozess, dem Standard- oder Advanced-Client mit der Anweisung JobStart() gestartet wurden. Bei Abfrage dieser Eigenschaft wird die Anzahl jedesmal neu ermittelt.



Es werden nur die Threads ermittelt, die aus dem Task oder Job erzeugt wurden. Threads, die von diesen Threads oder von anderen Tasks gestartet wurden, können nicht ermittelt werden.

Die Eigenschaft kann nur gelesen werden.

## Kontakt

**LclCurrDecimal**  Dezimaltrennzeichen für die  
Währungsdarstellung Typ alpha(4)

Siehe Liste, SysPropGet(), SysPropSet()

In dieser Eigenschaft wird das Dezimaltrennzeichen definiert. Es können maximal vier Zeichen angegeben werden.

Beispiel:

```
tHdLocale->spLclCurrDecimal # ',';
```

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclCurrFract  Anzahl der voreingestellten Nachkommastellen bei Währungsdarstellung

Typ int

Siehe [Liste](#), [SysPropGet\(\)](#), [SysPropSet\(\)](#)

In dieser Eigenschaft wird der Defaultwert für die Anzahl der Nachkommastellen eingetragen.

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option

FmtNumCurrency angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle

SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclCurrFractIntl



Anzahl der voreingestellten Nachkommastellen bei internationaler Währungsdarstellung

Typ **int**

Siehe [Liste](#), [SysPropGet\(\)](#), [SysPropSet\(\)](#)

In dieser Eigenschaft wird der Defaultwert für die Anzahl der Nachkommastellen für die internationale Währung eingetragen.

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option [FmtNumCurrencyIntl](#) angegeben wird.

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

## Kontakt

LclCurrGroup ┌ Gruppierung für die  
Währungsdarstellung Typ alpha(9)

Siehe Liste, SysPropGet(), SysPropSet()

In dieser Eigenschaft wird die Gruppierung bei größeren Zahlen festgelegt. Die angegebene Zeichenkette kann maximal neun Zeichen beinhalten. Die Anzahl der Ziffern, die zu einer Gruppe gehören, werden durch Semikolon getrennt angegeben. Durch die Angabe von '0' nach dem letzten Semikolon, wird die vorhergehende Gruppe für größere Zahlen verwendet.

**Beispiel:**

```
tHdlLocale->spLclCurrTSep # '.'; tHdlLocale->spLclCurrGroup # '3;0';
```

Diese Einstellungen führen zu folgender Darstellung:

```
'12345678' -> '12.345.678'
```

Es können auch Gruppen mit unterschiedlichen Längen angegeben werden:

```
tHdlLocale->spLclNumTSep # '.'; tHdlLocale->spLclNumGroup # '3;2;0';
```

Dies führt zu folgender Darstellung:

```
'12345678' -> '1.23.45.678'
```

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclCurrNegSignPos



Position des negativen Vorzeichens bei Währungsdarstellung

Typ int

Siehe Liste, SysPropGet(), SysPropSet()

In dieser Eigenschaft wird die Darstellung negativer Zahlen eingestellt. Folgende Werte können gesetzt werden:

Wert Darstellung	Beispiel
0 (<wert>)	'(12)'
1 <vorzeichen><wert>	'-12'
2 <vorzeichen><Leerzeichen><wert>	' - 12'
3 <wert><vorzeichen>	'12-'
4 <wert><Leerzeichen><vorzeichen>	'12 -'

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclCurrNegSpaceSep



Trennung des Währungssymbols vom Wert bei negativen Zahlen Typ int

Siehe [Liste](#), [SysPropGet\(\)](#), [SysPropSet\(\)](#)

Ist diese Eigenschaft auf den Wert 0 gesetzt, wird zwischen dem Wert und dem Währungssymbol kein Leerzeichen angezeigt. Soll ein Leerzeichen angezeigt werden, muss die Eigenschaft auf 1 gesetzt werden.

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option [FmtNumCurrency](#) oder [FmtNumCurrencyIntl](#) angegeben wird.

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

## Kontakt

---

**LclCurrNegSymPrec**

**Position des Währungssymbols**

Typ int  
Siehe Liste, SysPropGet(),

Wird in dieser Eigenschaft der Wert 0 eingetragen, wird das Währungssymbol hinter dem Wert dargestellt. Bei 1 wird das Symbol vor dem Wert angezeigt.

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclCurrPosSignPos



Position des positiven Vorzeichens bei Währungsdarstellung

Typ int

Siehe Liste, SysPropGet(), SysPropSet()

In dieser Eigenschaft wird die Position des positiven Vorzeichens definiert. Folgende Werte können eingetragen werden:

Wert Darstellung	Beispiel
0 (<wert>)	'(12)'
1 <vorzeichen><wert>	'+12'
2 <vorzeichen><Leerzeichen><wert>	' + 12'
3 <wert><vorzeichen>	'12+'
4 <wert><Leerzeichen><vorzeichen>	'12 +'

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

**LclCurrPosSpaceSep**  Trennung des Währungssymbols vom Wert Typ int

Siehe [Liste](#), [SysPropGet\(\)](#), [SysPropSet\(\)](#)

Ist diese Eigenschaft auf den Wert 0 gesetzt, wird zwischen dem Wert und dem Währungssymbol kein Leerzeichen angezeigt. Soll ein Leerzeichen angezeigt werden, muss die Eigenschaft auf 1 gesetzt werden.

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option [FmtNumCurrency](#) oder [FmtNumCurrencyIntl](#) angegeben wird.

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

## Kontakt

---

**LclCurrPosSymPrec**

**Position des Währungssymbols**

**Typ** int  
**Siehe** Liste, SysPropGet(),

**Wird in dieser Eigenschaft der Wert 0 eingetragen, wird das Währungssymbol hinter dem Wert dargestellt. Bei 1 wird das Symbol vor dem Wert angezeigt.**

**Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.**

**Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.**

## Kontakt

LclCurrSymbol

Währungssymbol

Typ alpha(7)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das Währungssymbol definiert. Es kann ein Währungssymbol mit bis zu sieben Zeichen angegeben werden.

Beispiel:

```
tHdILocale->spLclCurrSymbol # '€';
```

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

### LclCurrSymbolIntl

Internationales Währungssymbol

Typ [alpha\(7\)](#)

[Liste](#), [SysPropGet\(\)](#),

Siehe

In dieser Eigenschaft kann das internationale Währungssymbol eingestellt werden.

Das Währungssymbol kann bis zu sieben Zeichen lang sein.

**Beispiel:**

```
tHdlLocale->spLclCurrSymbolIntl # 'EUR';
```

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option

[FmtNumCurrencyIntl](#) angegeben wird.

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle

[SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

## Kontakt

---

LclCurrTSep  $\sqsubset$  Tausendertrennzeichen für die  
Währungsdarstellung Typ alpha(4)

Siehe Liste, SysPropGet(), SysPropSet()

In dieser Eigenschaft wird das Tausendertrennzeichen definiert. Es können maximal vier Zeichen angegeben werden.

Beispiel:

```
tHdLocale->spLclCurrTSep # '.';
```

Diese Einstellung wird verwendet, wenn bei der Konvertierung die Option FmtNumCurrency oder FmtNumCurrencyIntl angegeben wird.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateDayLZero

Tag mit führender Null

Typ int

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird angegeben, ob die Tagesangabe im langen Datumsformat mit (1) oder ohne (0) führende Null dargestellt wird.

Wird die Eigenschaft LclDateLFormat gesetzt, muss diese Eigenschaft auf den entsprechenden Wert gesetzt werden.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateDayN

Namen der Wochentage

Typ alpha(23)[7]

Siehe Liste, SysPropGet(),

In dieser Eigenschaft befinden sich die Namen der Wochentage. Es können sieben Wochentage mit je bis zu 23 Zeichen angegeben werden. Bei der Zuweisung oder beim Auslesen muss die Position mitangegeben werden.

Die angegebenen Namen werden bei der Ausgabe im langen oder kurzen Datumsformat verwendet, wenn in der Eigenschaft LclDateSFormat oder LclDateLFormat dddd angegeben wird.

Beispiele:

```
// ZuweisungtHdlLocale->spLclDateDayN(1) # 'Montag'; tHdlLocale->spLclDateDayN(2) # 'Dienstag'; tH
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateDayNS

**Kurznamen der Wochentage**

**Typ** alpha(7)[7]  
Liste, SysPropGet(),  
**Siehe**

In dieser Eigenschaft befinden sich die Namen der Wochentage. Es können sieben Wochentage mit je bis zu 7 Zeichen angegeben werden. Bei der Zuweisung oder beim Auslesen muss die Position mitangegeben werden.

Die angegebenen Namen werden bei der Ausgabe im langen oder kurzen Datumsformat verwendet, wenn in der Eigenschaft LclDateSFormat oder LclDateLFormat ddd angegeben wird.

**Beispiel:**

```
// ZuweisungtHdlLocale->spLclDateDayNS(1) # 'Mo'; tHdlLocale->spLclDateDayNS(2) # 'Di'; tHdlLocale
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateLFormat

Langes Datumsformat

Typ alpha(79)

Liste,

Siehe SysPropGet(),  
SysPropSet()

In dieser Eigenschaft wird das lange Datumsformat festgelegt. Wird das Format geändert, müssen die Eigenschaften LclDateLOrder und LclDateMonthLZero an das neue Format angepasst werden.

Der Formatstring kann bis zu 79 Zeichen lang sein und folgende Zeichen beinhalten:

Wert	Darstellung	Beispiel
'd'	Nummer des Tages (ein oder zweistellig)	'29'
'dd'	Nummer des Tages (zweistellig)	'29'
'ddd'	Kurzschriftweise des Tages (Eigenschaft LclDateDayNS)	'Mo'
'dddd'	Wochentag (Eigenschaft LclDateDayN)	'Montag'
'M'	Nummer des Monats (ein- oder zweistellig)	'7'
'MM'	Nummer des Monats (zweistellig)	'07'
'MMM'	Kurzschriftweise des Monats (Eigenschaft LclDateMonthNS)	'Jul'
'MMMM'	Monat (Eigenschaft LclDateMonthN)	'Juli'
'y'	Letzte Ziffer des Jahres	'2'
'yy'	Zweistellige Jahresanzeige	'02'
'yyyy'	Vierstellige Jahresanzeige	'2002'

Die folgenden Beispiele beziehen sich auf das Datum 29.07.2002.

```
// Format: Montag 29. Juli 2002tHdlLocale->spLclDateLFormat # 'dddd d. MMMM yyyy'; tHdlLocale->sp
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateLOrder

#### Datumsreihenfolge

Typ int

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird die Reihenfolge der Angaben in einem Datumswert definiert.

Folgende Werte können angegeben werden:

**0 Monat - Tag - Jahr**

**1 Tag - Monat - Jahr**

**2 Jahr - Monat - Tag**

Wird die Eigenschaft LclDateLFormat gesetzt, muss diese Eigenschaft auf die gleiche Reihenfolge gesetzt werden.

**Beispiel:**

```
tHdlLocale->spLclDateLFormat # 'yyyy-MM-dd'; tHdlLocale->spLclDateSep # '-'; // Anpassen der ander
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle

SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

---

**LclDateMonthLZero**

**Monat mit führender Null**

**Typ** int

**Siehe** Liste, SysPropGet(),

In dieser Eigenschaft wird angegeben, ob die Monatsangabe im langen Datumsformat mit (1) oder ohne (0) führende Null dargestellt wird.

Wird die Eigenschaft LclDateLFormat gesetzt, muss diese Eigenschaft auf den entsprechenden Wert gesetzt werden.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateMonthN

#### Namen der Monate

Typ alpha(23)[12]

Liste,

Siehe SysPropGet(),  
SysPropSet()

In dieser Eigenschaft befinden sich die langen Namen der Monate. Es können zwölf Monate mit je bis zu 23 Zeichen angegeben werden. Bei der Zuweisung oder beim Auslesen muss die Position mit angegeben werden.

Die angegebenen Namen werden bei der Ausgabe im langen oder kurzen Datumsformat verwendet, wenn in der Eigenschaft LclDateSFormat oder LclDateLFormat MMMM angegeben wird.

Beispiel:

```
// Zuweisung tHdlLocale->spLclDateMonthN(1) # 'Januar'; tHdlLocale->spLclDateMonthN(2) # 'Februar';
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateMonthNS

#### Kurznamen der Monate

Typ alpha(7)[12]  
Liste, SysPropGet(),  
Siehe

In dieser Eigenschaft befinden sich die kurzen Namen der Monate. Es können zwölf Monate mit je bis zu sieben Zeichen angegeben werden. Bei der Zuweisung oder beim Auslesen muss die Position mitangegeben werden.

Die angegebenen Namen werden bei der Ausgabe im langen oder kurzen Datumsformat verwendet, wenn in der Eigenschaft LclDateSFormat oder LclDateLFormat MMM angegeben wird.

Beispiel:

```
// ZuweisungtHdlLocale->spLclDateMonthNS(1) # 'Jan'; tHdlLocale->spLclDateMonthNS(2) # 'Feb'; tHdl
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclDateSCentury

Darstellung des Jahres

Typ int

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird definiert, mit wievielen Stellen das Jahr im kurzen Datumsformat angezeigt wird. Die Darstellung wird in der Eigenschaft LclDateSFormat gesetzt. Diese Eigenschaft muss entsprechend dem Format angepasst werden.

Wird in dieser Eigenschaft der Wert 0 angegeben, wird eine zweistellige Jahresangabe angenommen.

Bei 1 wird eine vierstellige Jahresangabe angenommen.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle

SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

**LclDateSep**

**Datumsseparator**

**Typ** alpha(4)

Liste,

**Siehe** SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das Trennzeichen für das Datum gesetzt. Das Trennzeichen kann maximal vier Zeichen lang sein.

Wird die Eigenschaft LclDateSFormat gesetzt, muss diese Eigenschaft auf den gleichen Separator gesetzt werden.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclDateSFormat

#### Kurzes Datumsformat

Typ [alpha\(79\)](#)

[Liste](#),

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

[SysPropSet\(\)](#)

In dieser Eigenschaft wird das kurze Datumsformat festgelegt. wird das Format geändert, müssen die Eigenschaften [LclDateSep](#), [LclDateSOrder](#) und [LclDateSCentury](#) an das neue Format angepasst werden.

Der Formatstring kann bis zu 79 Zeichen lang sein und folgende Zeichen beinhalten:

Wert	Darstellung	Beispiel
'd'	Nummer des Tages (ein oder zweistellig)	'29'
'dd'	Nummer des Tages (zweistellig)	'29'
'ddd'	Kurzschriftweise des Tages (Eigenschaft LclDateDayNS)	'Mo'
'dddd'	Wochentag (Eigenschaft LclDateDayN)	'Montag'
'M'	Nummer des Monats (ein- oder zweistellig)	'7'
'MM'	Nummer des Monats (zweistellig)	'07'
'MMM'	Kurzschriftweise des Monats (Eigenschaft LclDateMonthNS)	'Jul'
'MMMM'	Monat (Eigenschaft LclDateMonthN)	'Juli'
'y'	Letzte Ziffer des Jahres	'2'
'yy'	Zweistellige Jahresanzeige	'02'
'yyyy'	Vierstellige Jahresanzeige	'2002'

Die folgenden Beispiele beziehen sich auf das Datum 29.07.2002.

```
tHdlLocale->spLclDateSFormat # 'yy-MM-dd'; // 02-07-29// Anpassen der anderen Eigenschaften tHdlLo
```

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

## Kontakt

**LclDateSOrder**

Datumsreihenfolge

Typ int

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird die Reihenfolge der Angaben in einem Datumswert definiert.

Folgende Werte können angegeben werden:

**0 Monat - Tag - Jahr**

**1 Tag - Monat - Jahr**

**2 Jahr - Monat - Tag**

Wird die Eigenschaft LclDateSFormat gesetzt, muss diese Eigenschaft auf die gleiche Reihenfolge gesetzt werden.

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle

SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

### LclNumDecimal

Dezimaltrennzeichen

Typ alpha(4)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das Dezimaltrennzeichen definiert. Es können maximal vier Zeichen angegeben werden.

Beispiel:

```
tHdILocale->spLclNumDecimal # ',';
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

---

LclNumFract  Anzahl der voreingestellten  
Nachkommastellen Typ int

Siehe [Liste](#), [SysPropGet\(\)](#), [SysPropSet\(\)](#)

In dieser Eigenschaft wird der Defaultwert für die Anzahl der Nachkommastellen eingetragen.

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

## Kontakt

LclNumGroup



Gruppierung

Typ alpha(9)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird die Gruppierung bei größeren Zahlen festgelegt. Die angegebene Zeichenkette kann maximal neun Zeichen beinhalten. Die Anzahl der Ziffern, die zu einer Gruppe gehören, werden durch Semikolon getrennt angegeben. Durch die Angabe von 0 nach dem letzten Semikolon, wird die vorhergehende Gruppe für größere Zahlen verwendet.

**Beispiel:**

```
tHdLocale->spLclNumTSep # '.';tHdLocale->spLclNumGroup # '3;0';
```

Diese Einstellungen führen zu folgender Darstellung:

```
'12345678' -> '12.345.678'
```

Es können auch Gruppen mit unterschiedlichen Längen angegeben werden:

```
tHdLocale->spLclNumTSep # '.';tHdLocale->spLclNumGroup # '3;2;0';
```

Dies führt zu folgender Darstellung:

```
'12345678' -> '1.23.45.678'
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

---

**LclNumLZero**

**Darstellung mit führender Null**

**Typ** int  
**Siehe** Liste, SysPropGet(),

**Wird in dieser Eigenschaft 0 eingetragen, wird keine führende 0 dargestellt. Durch Angabe von 1 erfolgt die Darstellung mit einer führenden 0.**

**Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.**

## Kontakt

LclNumNegMode Darstellung

negativer Zahlen Typ int

Siehe Liste, SysPropGet(),  
SysPropSet()

In dieser Eigenschaft wird die Darstellung negativer Zahlen eingestellt. Folgende Werte können gesetzt werden:

Wert Darstellung	Beispiel
0 <wert>	'(12)'
1 <vorzeichen><wert>	'-12'
2 <vorzeichen><Leerzeichen><wert>	'- 12'
3 <wert><vorzeichen>	'12-'
4 <wert><Leerzeichen><vorzeichen>	'12 -'

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

### LclNumNegSign

Negatives Vorzeichen

Typ alpha(5)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das negative Vorzeichen definiert. Es können maximal fünf Zeichen angegeben werden.

Beispiel:

```
tHdLocale->spLclNumNegSign # '-' ;
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

### LclNumPosSign

Positives Vorzeichen

Typ [alpha\(2\)](#)

[Liste](#),

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

[SysPropSet\(\)](#)

In dieser Eigenschaft wird das positive Vorzeichen definiert. Es können maximal zwei Zeichen angegeben werden.

Beispiel:

```
tHdILocale->spLclNumPosSign # '+';
```

Wird diese Eigenschaft gesetzt, erscheint das Vorzeichen vor positiven Zahlen im Währungsformat. Sollen positive Zahlen ohne Vorzeichen dargestellt werden, muss diese Eigenschaft auf den Leerstring ('') gesetzt werden.



Diese Eigenschaft wird nur ausgewertet, wenn eine Zahl mit [CnvAF\(\)](#) oder [CnvAM\(\)](#) und der Option [FmtNumCurrency](#) in eine [Zeichenkette](#) umgewandelt wird.

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

### LclNumTSep

Tausendertrennzeichen

Typ alpha(4)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das Tausendertrennzeichen definiert. Es können maximal vier Zeichen angegeben werden.

Beispiel:

```
tHdLocale->spLclNumTSep # '.';
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclTimeHourMode

Zeitformat

Typ int

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft ist das Zeitformat definiert. Ist hier 0 angegeben, wird die Uhrzeit im 12-Stunden-Format, bei 1 im 24-Stunden-Format angegeben.

Beispiel:

```
tHdILocale->spLclTimeHourMode # 1; // 24-Stunden-Format
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclTimeLFormat



Langes Zeitformat

Typ alpha(79)

Liste,

Siehe SysPropGet(),

SysPropSet(),

LclTimeSFormat

In dieser Eigenschaft wird das lange Zeitformat festgelegt. Wird das Format geändert, müssen gegebenenfalls die Eigenschaften LclTimeHourMode, LclTimeLZero, LclTimeMarkPos und LclTimeSep an das neue Format angepasst werden.

Der Formatstring kann bis zu 79 Zeichen lang sein und folgende Zeichen beinhalten:

Wert Darstellung	Beispiel
'h' Stunde im 12-Stundenformat (ein oder zweistellig)	'2'
'hh' Stunde im 12-Stundenformat (zweistellig)	'02'
'H' Stunde im 24-Stundenformat (ein oder zweistellig)	'14'
'HH' Stunde im 24-Stundenformat (zweistellig)	'14'
'm' Minute (ein- oder zweistellig)	'6'
'mm' Minute (zweistellig)	'06'
's' Sekunde (ein- oder zweistellig)	'35'
'ss' Sekunde (zweistellig)	'35'
'n' Hunderstel-Sekunde (ein- oder zweistellig)	'42'
'nn' Hunderstel-Sekunde (zweistellig)	'42'
't' Symbol (ein Zeichen)	'P'
'tt' Symbol (zwei Zeichen)	'PM'
'T' komplettes Symbol (bis zu neun Zeichen)	'PM'

Die Beispiele beziehen sich auf die Uhrzeit 14:06:35.42.

Beispiel:

```
tHdlLocale->spLclTimeLFormat # 'HH:mm:ss.nn'; // 14:06:35.42tHdlLocale->spLclTimeSep # ':'; // An
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.



Bis einschließlich Windows Vista hat diese Eigenschaft standardmäßig den gleichen Wert, wie LclTimeSFormat.

### LclTimeLZero

Stunden mit führender Null

Typ int  
Siehe Liste, SysPropGet(),

In dieser Eigenschaft wird angegeben, ob die Stunden mit einer führenden 0 ausgegeben werden (1) oder nicht (0).

Beispiel:

```
tHdlLocale->spLclTimeLZero # 1;
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclTimeMarkPos

Position des Zeitsymbols

Typ int  
Siehe Liste, SysPropGet(),

Bei der Anzeige im 12-Stunden-Format wird ein Symbol zur Anzeige des Vor- bzw. Nachmittags verwendet. Die Position befindet sich entweder hinter der Uhrzeit (LclTimeMarkPos = 0) oder vor der Uhrzeit (LclTimeMarkPos = 1).

Beispiel:

```
tHdLocale->spLclTimeMarkPos # 1;
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

LclTimeSep



Zeitseparatör

Typ alpha(4)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft steht das Trennzeichen der Uhrzeit. Das Trennzeichen kann bis zu vier Zeichen lang sein.

Beispiel:

```
tHd1Locale->spLclTimeSep # ':';
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

### LclTimeSepAM

Symbol für Vormittag

Typ alpha(9)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das Symbol für den Vormittag angegeben. Das Symbol wird nur dargestellt, wenn die Uhrzeit im 12-Stunden-Format angegeben wird. Das Symbol kann bis zu neun Zeichen lang sein.

Beispiel:

```
tHdlLocale->spLclTimeSepAM # 'AM';
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

### LclTimeSepPM

Symbol für Nachmittag

Typ alpha(9)

Liste,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft wird das Symbol für den Nachmittag angegeben. Das Symbol wird nur dargestellt, wenn die Uhrzeit im 12-Stunden-Format angegeben wird. Das Symbol kann bis zu neun Zeichen lang sein.

Beispiel:

```
tHdlLocale->spLclTimeSepPM # 'PM';
```

Die Eigenschaft kann nach dem Laden eines Locale-Objekts mit LocaleLoad() über die Befehle SysPropGet() und SysPropSet() gelesen bzw. geändert werden.

## Kontakt

### LclTimeSFormat

#### Kurzes Zeitformat

Typ [alpha\(79\)](#)

[Liste](#),

Siehe [SysPropGet\(\)](#),  
[SysPropSet\(\)](#)

In dieser Eigenschaft wird das kurze Zeitformat festgelegt. Wird das Format geändert, müssen gegebenenfalls die Eigenschaften [LclTimeHourMode](#), [LclTimeLZero](#), [LclTimeMarkPos](#) und [LclTimeSep](#) an das neue Format angepasst werden.

Der Formatstring kann bis zu 79 Zeichen lang sein und folgende Zeichen beinhalten:

Wert Darstellung	Beispiel
'h' Stunde im 12-Stundenformat (ein oder zweistellig)	'2'
'hh' Stunde im 12-Stundenformat (zweistellig)	'02'
'H' Stunde im 24-Stundenformat (ein oder zweistellig)	'14'
'HH' Stunde im 24-Stundenformat (zweistellig)	'14'
'm' Minute (ein- oder zweistellig)	'6'
'mm' Minute (zweistellig)	'06'
's' Sekunde (ein- oder zweistellig)	'35'
'ss' Sekunde (zweistellig)	'35'
'n' Hunderstel-Sekunde (ein- oder zweistellig)	'42'
'nn' Hunderstel-Sekunde (zweistellig)	'42'
't' Symbol (ein Zeichen)	'P'
'tt' Symbol (zwei Zeichen)	'PM'
'T' komplettes Symbol (bis zu neun Zeichen)	'PM'

Die Beispiele beziehen sich auf die Uhrzeit 14:06:35.42.

Beispiel:

```
tHdlLocale->spLclTimeSFormat # 'HH:mm:ss.nn'; // 14:06:35.42tHdlLocale->spLclTimeSep # ':'; // An
```

Die Eigenschaft kann nach dem Laden eines [Locale](#)-Objekts mit [LocaleLoad\(\)](#) über die Befehle [SysPropGet\(\)](#) und [SysPropSet\(\)](#) gelesen bzw. geändert werden.

-  Bis einschließlich Windows Vista hat diese Eigenschaft standardmäßig den gleichen Wert, wie [LclTimeLFormat](#).
-  Bis zur Version 5.7.08 lieferte die Eigenschaft LclTimeSFormat unter Windows 7 und Windows 8 immer das lange Zeitformat (siehe [LclTimeLFormat](#)).

## Kontakt

Len Aktuelle Datenmenge im Memory-Objekt

Typ int

Siehe Liste, SysPropGet(), SysPropSet()

In dieser Eigenschaft steht die aktuelle Datenmenge im Speicher des Objekts. Der Minimalwert ist 0, der Maximalwert entspricht Size. Der Wert bedeutet, dass die Bytes von Position 1 bis zu dem angegebenen Wert im Speicher belegt sind. Der Wert wird bei bestimmten Operationen verwendet. Bei einigen Befehlen kann bei der Datenlänge die Konstante MemDataLen übergeben werden, wodurch dann dieser Wert verwendet wird.

### LogicalProcessors

Anzahl der logischen Prozessoren

Typ int

Siehe Liste, Objekte, SysPropGet()

Über diese Eigenschaft kann die Anzahl der logischen Prozessoren des Rechners ermittelt werden, auf dem der Client läuft.

Beispiel:

```
tProcessors # _Sys->spLogicalProcessors;
```

## Kontakt

### Eigenschaften von System-Objekten - M bis O

In dieser Liste sind alle Eigenschaften von M bis O von System-Objekten aufgeführt. [Liste sortiert](#)

[nach Gruppen](#),

Siehe [Alphabetische](#)

[Liste aller](#)

[Eigenschaften](#)

- [Method](#)
- [Modified](#)
- [ModifiedUser](#)
- [Name](#)
- [ObjectsGDI](#)
- [ObjectsGDILimit](#)
- [ObjectsUser](#)
- [ObjectsUserLimit](#)
- [OdbcClmBufferLen](#)
- [OdbcClmCatalog](#)
- [OdbcClmDataDefault](#)
- [OdbcClmDataType](#)
- [OdbcClmDataTypeName](#)
- [OdbcClmDecimalDigits](#)
- [OdbcClmName](#)
- [OdbcClmNullable](#)
- [OdbcClmNumPrecRadix](#)
- [OdbcClmOrdinal](#)
- [OdbcClmRemarks](#)
- [OdbcClmSchema](#)
- [OdbcClmSize](#)
- [OdbcClmTable](#)
- [OdbcConCfcOdbcApi](#)
- [OdbcConCfcOdbcSql](#)
- [OdbcConCfcSagCli](#)
- [OdbcConDatabaseName](#)
- [OdbcConDbmsName](#)
- [OdbcConDbmsVersion](#)
- [OdbcConDriverName](#)
- [OdbcConDriverOdbcVersion](#)
- [OdbcConDriverVersion](#)
- [OdbcConDSN](#)
- [OdbcConOdbcVersion](#)
- [OdbcConReadOnly](#)
- [OdbcConServerName](#)
- [OdbcConUserName](#)
- [OdbcErrSqlMessage](#)
- [OdbcErrSqlNativeCode](#)
- [OdbcErrSqlResult](#)
- [OdbcErrSqlState](#)
- [OdbcResCountClm](#)
- [OdbcResCountRow](#)
- [OdbcTblCatalog](#)

## Kontakt

- OdbcTblName
- OdbcTblRemarks
- OdbcTblSchema
- OdbcTblType

## Kontakt

---

### Method

#### Methode der HTTP-Anfrage

Typ alpha(15)

Liste, Objekte,

Siehe SysPropGet(),

SysPropSet()

Die Eigenschaft beinhaltet die Methode der HTTP-Anfrage. Wird ein neues HTTP-Objekt mit HttpOpen( HttpSendRequest, ...) erzeugt, ist die Eigenschaft mit 'GET' vorbelegt. Die Eigenschaft darf nur auf gültige Methoden gesetzt werden.

Die Eigenschaft ist bei den Typen HttpSendResponse und HttpRecvResponse nicht vorhanden. Bei dem Objekt-Typ HttpRecvRequest kann sie nur gelesen werden.

## Kontakt

---

### Modified

Zeitpunkt der letzten Änderung

Typ caltime

Siehe Liste, Objekte,

Diese Eigenschaft enthält Datum und Uhrzeit der letzten Änderung des Objekts. Für Validierungselemente kann die Eigenschaft gelesen und geschrieben werden. Von allen anderen Objekten ist sie nur lesbar.

Der Zeitpunkt wird vom System neu gesetzt, wenn eine Datei in dieses Objekt importiert oder der Inhalt gelöscht wird. Änderungen an den Eigenschaften des Objekts führen nicht zu einem neuen Setzen des Änderungszeitpunktes.

Bei Validierungselementen wird diese Eigenschaft im Validierungs-Editor automatisch aktualisiert, wenn das Objekt nach der letzten Validierung verändert wurde und der Validierungsstatus erneut ermittelt wird. Hierbei bezieht sich die Eigenschaft auf das Referenzobjekt in der Datenbank.

## Kontakt

ModifiedUser



Benutzer, der das Objekt zuletzt geändert hat Typ  
alpha(20)

Siehe Liste, Objekte, Modified

Diese Eigenschaft enthält den Namen des Benutzers, der das Objekt zuletzt geändert hat. Für Validierungselemente kann die Eigenschaft gelesen und geschrieben werden. Von allen anderen Objekten ist sie nur lesbar.

Der Benutzer wird vom System neu gesetzt, wenn eine Datei in dieses Objekt importiert oder der Inhalt gelöscht wird. Änderungen an den Eigenschaften des Objekts führen nicht zu einem neuen Setzen des Änderungsbewerters.

Bei Validierungselementen wird diese Eigenschaft im Validierungs-Editor automatisch aktualisiert, wenn das Objekt nach der letzten Validierung verändert wurde und der Validierungsstatus erneut ermittelt wird. Hierbei bezieht sich die Eigenschaft auf das Referenzobjekt in der Datenbank.

## Kontakt

Name (Systemeigenschaft)  —

Name des Systemobjekts

Typ alpha(60) oder  
alpha(8192)

Siehe Liste, Objekte,  
SysPropGet()

In dieser Eigenschaft steht der Name des Objekts. Die Eigenschaft kann bei binären Objekten und beim Objekt TapiDevice nur gelesen werden. Bei den Binären Objekten wird der Name beim Erstellen des Verzeichnisses oder Objekts vergeben.

Bei den dynamischen Strukturen kann der Name eines Item-Objekts gesetzt werden, bis das Item in eine Baumstruktur integriert wurde.

Der Name von Item-Objekten in dynamischen Strukturen kann bis zu 8192 Zeichen lang sein. Die Länge der Namen von anderen Objekten ist auf 60 Zeichen beschränkt.

### ObjectsGDI

Verbrauch der GDI-Objekte ermitteln

Typ [int](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Über diese Eigenschaft wird der Verbrauch der GDI-Objekte ermittelt. Das Limit kann mit der Eigenschaft [ObjectsGDILimit](#) ermittelt werden.

Beispiel:

```
tGDIObjects      # _Sys->spObjectsGDI;tGDIObjectsLimit      # _Sys->spObjectsGDILimit;tGDIObjectsP
```



Diese Eigenschaft kann nur unter Windows ermittelt werden. Unter Linux wird 0 zurückgegeben.

### ObjectsGDILimit

Limit der GDI-Objekte ermitteln

Typ int

Siehe Liste, Objekte, SysPropGet()

Über diese Eigenschaft wird das Limit der GDI-Objekte ermittelt. Der Verbrauch kann mit der Eigenschaft ObjectsGDI ermittelt werden.

Beispiel:

```
tGDIObjects      # _Sys->spObjectsGDI;tGDIObjectsLimit      # _Sys->spObjectsGDILimit;tGDIObjectsP
```



Diese Eigenschaft kann nur unter Windows ermittelt werden. Unter Linux wird 0 zurückgegeben.

## Kontakt

---

### ObjectsUser

Verbrauch der Benutzer-Objekte ermitteln

Typ [int](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Über diese Eigenschaft wird der Verbrauch der Benutzer-Objekte ermittelt. Das Limit kann mit der Eigenschaft [ObjectsUserLimit](#) ermittelt werden.

Beispiel:

```
tUserObjects      # _Sys->spObjectsUser;tUserObjectsLimit      # _Sys->spObjectsUserLimit;tUserObj
```



Diese Eigenschaft kann nur unter Windows ermittelt werden. Unter Linux wird 0 zurückgegeben.

## Kontakt

---

### ObjectsUserLimit

Limit der Benutzer-Objekte ermitteln

Typ [int](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Über diese Eigenschaft wird das Limit der Benutzer-Objekte ermittelt. Der Verbrauch kann mit der Eigenschaft [ObjectsUser](#) ermittelt werden.

Beispiel:

```
tUserObjects      # _Sys->spObjectsUser;tUserObjectsLimit      # _Sys->spObjectsUserLimit;tUserObj
```



Diese Eigenschaft kann nur unter Windows ermittelt werden. Unter Linux wird 0 zurückgegeben.

## Kontakt

OdbcClmBufferLen Größe

der Spalte in Byte Typ int

Siehe Liste, Objekte

Die Eigenschaft gibt die Größe der Spalte in Byte zurück. Eine Spalte vom Datentyp OdbcTypeSmallInt hat zum Beispiel eine Größe von 2 Byte.

## Kontakt

OdbcClmCatalog

**Katalog der Spalte**

**Typ alpha**

**Siehe Liste, Objekte**

**In dieser Eigenschaft wird der Name des Katalogs (der Datenbank) ausgegeben, zu der die Spalte gehört.**

## Kontakt

OdbcClmDataDefault

Vorgabewert der Spalte

Typ alpha

Siehe Liste, Objekte

In dieser Eigenschaft ist der Vorgabewert der Spalte abgelegt. Handelt es sich bei dem Datentyp der Spalte nicht um eine Zeichenkette, ist der entsprechende Vorgabewert in eine Zeichenkette gewandelt.

### OdbcClmDataType

Datentyp der Spalte

Typ int

Siehe Liste, Objekte

In dieser Eigenschaft wird der Datentyp der Spalte ausgegeben. Der Wert kann mit folgenden Konstanten verglichen werden:

Datentyp	ODBC-Datentyp	Datentyp in CONZEPT 16
_OdbcTypeBit	Bitwert	nicht verfügbar
_OdbcTypeChar	Zeichenkette fester Länge	<u>alpha</u>
_OdbcTypeVarChar	Zeichenkette variabler Länge	<u>alpha</u>
_OdbcTypeLongVarChar	Text	<u>alpha(8192)</u> Der Text wird nach 4096 Zeichen abgeschnitten. Alternativ kann ein <u>Memory-Objekt</u> mit dem Datentyp <u>handle</u> angegeben werden.
_OdbcTypeWChar	UTF16-Zeichenkette fester Länge	nicht verfügbar
_OdbcTypeWVarChar	UTF16-Zeichenkette variabler Länge	nicht verfügbar
_OdbcTypeWLongVarChar	UTF16-Text	nicht verfügbar
_OdbcTypeNumeric	numerischer Datentyp	nicht verfügbar
_OdbcTypeTinyInt	vorzeichenbehafteter Integer	(-128 ... +127) byte <u>Muss</u> umgerechnet werden.
_OdbcTypeSmallInt	vorzeichenbehafteter 16-bit Integer	<u>word</u> Muss umgerechnet werden.
_OdbcTypeInteger	vorzeichenbehafteter 32-bit Integer	<u>int</u>
_OdbcTypeBigInt	vorzeichenbehafteter 64-bit Integer	<u>bignint</u>
_OdbcTypeDecimal	Dezimal	nicht verfügbar
_OdbcTypeFloat	32-bit Fließkomma	nicht verfügbar
_OdbcTypeReal	32-bit Fließkomma	nicht verfügbar
_OdbcTypeDouble	64-bit Fließkomma	<u>float</u>
_OdbcTypeDate	Datumswert	<u>date</u>
_OdbcTypeTime	Zeitwert	<u>time</u>
_OdbcTypeTimestamp	Timestamp	<u>caltime</u>
_OdbcTypeVarBinary	binäre Daten	nicht verfügbar
_OdbcTypeLongVarBinary	binäre Daten	nicht verfügbar



Die vorzeichenbehafteten Datentypen OdbcTypeTinyInt und OdbcTypeSmallInt werden auf die vorzeichenlosen Datentypen byte und word gemappt. Negative Werte werden in der oberen Hälfte des Werteraumes abgebildet. Der Betrag des Wertes kann durch die folgende Funktion ermittelt werden.

Beispiel:

## Kontakt

```
tSign # 1;if (tValue > 32767) // 127 für TinyInt{ tSign # -1; tAbs # 65535 - tValue + 1; // 255
```

## Kontakt

OdbcClmDataTypeName Ausgeschriebener

Name des Datentyps Typ alpha

Siehe Liste, Objekte

Diese Eigenschaft enthält den Datentyp der Spalte in lesbarer Form.

## Kontakt

OdbcClmDecimalDigits

Anzahl der Dezimalstellen Typ

int

Siehe [Liste, Objekte](#)

Diese Eigenschaft gibt die Anzahl der Dezimalziffern oder Bits zurück (siehe [OdbcClmNumPrecRadix](#)). Der Wert ist 0, wenn der Datentyp keine Dezimalstellen hat (zum Beispiel bei Zeichenketten).

## Kontakt

**OdbcClmName**

Name der Spalte

Typ alpha

Liste,

Siehe Objekte

In der Eigenschaft steht der Name der Spalte.

## Kontakt

**OdbcClmNullable**

**Wert NULL zulässig**

**Typ int**

**Siehe Liste, Objekte**

Über diese Eigenschaft kann ermittelt werden, ob der Wert NULL in der Spalte zulässig ist.

Die Eigenschaft kann folgende Werte enthalten:

- **0 - Der Wert der Spalte kann nicht auf NULL gesetzt werden.**
- **1 - Der Wert der Spalte kann auf NULL gesetzt werden.**
- **2 - Es kann nicht ermittelt werden, ob die Spalte auf den Wert NULL gesetzt werden kann.**

## Kontakt

OdbcClmNumPrecRadix



Anzahl der Stellen in Bits oder Dezimalziffern Typ int

Siehe Liste, Objekte

In dieser Eigenschaft kann ermittelt werden, welche Angabe in den Eigenschaften OdbcClmSize und OdbcClmDecimalDigits gemacht wurden. Folgende Werte können in der Eigenschaft zurückgegeben werden:

- 0 Die Spalte ist nicht von einem numerischen Typ.
- 2 Die Angaben in den Eigenschaften OdbcClmSize und OdbcClmDecimalDigits beziehen sich auf die Anzahl von Bits.
- 10 Die Angaben in den Eigenschaften OdbcClmSize und OdbcClmDecimalDigits beziehen sich auf die Anzahl von Dezimalziffern.

## Kontakt

OdbcClmOrdinal



Nummer der Spalte in der Tabelle

Typ int

Siehe Liste, Objekte

In dieser Eigenschaft kann die Nummer der Spalte in der Tabelle abgefragt werden.

Die Nummerierung beginnt mit 1.

## Kontakt

OdbcClmRemarks

Beschreibung der Spalte Typ

alpha

Siehe Liste, Objekte

In der Eigenschaft wird die Beschreibung der Spalte ausgegeben.

## Kontakt

OdbcClmSchema

Schema der Spalte

Typ alpha

Siehe Liste, Objekte

In der Eigenschaft wird der Name des Schemas der Spalte ausgegeben, sofern der entsprechende Treiber dies unterstützt.

## Kontakt

OdbcClmSize



Größe der Spalte in Zeichen

Typ [int](#)

Siehe [Liste](#), [Objekte](#)

Diese Eigenschaft enthält die Größe der Spalte in Ziffern oder Bits (siehe [OdbcClmNumPrecRadix](#)). Eine Spalte vom Datentyp [OdbcTypeSmallInt](#) hat zum Beispiel eine Größe von 5 Dezimalziffern.

## Kontakt

OdbcClmTable



Name der Tabelle

Typ alpha

Siehe Liste, Objekte

In der Eigenschaft steht der Name der Tabelle zu der die Spalte gehört.

**OdbcConCfcOdbcApi**

**Grad der ODBC-Konformität**

**Typ** int

**Siehe** Liste, Objekte

In dieser Eigenschaft steht der Grad der ODBC-Konformität. Der Wert kann mit folgenden Konstanten verglichen werden:

- \_OdbcCfcApiNone

Die Datenquelle ist nicht ODBC-Konform.

- \_OdbcCfcApiLevel1

Die Datenquelle ist Level 1-Konform.

- \_OdbcCfcApiLevel2

Die Datenquelle ist Level 2-Konform.

## Kontakt

OdbcConCfcOdbcSql

**SQL-Sprachumfang**

Typ int

Siehe Liste, Objekte

In der Eigenschaft ist der unterstützte Sprachumfang der Datenquelle verzeichnet.

Der Wert der Eigenschaft kann mit folgenden Konstanten verglichen werden:

- \_OdbcCfcSqlMinimum

Minimaler Sprachumfang

- \_OdbcCfcSqlCore

Kern-Sprachumfang

- \_OdbcCfcSqlExtended

Erweiterter Sprachumfang

## Kontakt

OdbcConCfcSagCli

Konformität zur SAG

Typ int

Siehe Liste, Objekte

In dieser Eigenschaft steht ein Wert, der die Konformität zur SQL Access Group (SAG) CLI Spezifikation ausgegeben.

## Kontakt

OdbcConDatabaseName

Name der Datenbank

Typ alpha

Siehe Liste, Objekte

In der Eigenschaft steht der Name der Datenbank aus der Datenquelle.

## Kontakt

OdbcConDbmsName



Name des Datenbank Management Systems

Typ alpha

Siehe Liste, Objekte

In der Eigenschaft steht der Name des Datenbank Management Systems.

## Kontakt

OdbcConDbmsVersion



Version des Datenbank Management Systems

Typ [alpha](#)

Siehe [Liste, Objekte](#)

In der Eigenschaft steht die Version des Datenbank Management Systems.

## Kontakt

**OdbcConDriverName**

Name des ODBC-Treibers

Typ alpha

Siehe Liste, Objekte

In der Eigenschaft steht der Name des ODBC-Treibers, mit dem auf die Datenquelle zugegriffen wird.

## Kontakt

OdbcConDriverOdbcVersion  Unterstützte

**ODBC-Version des Treibers Typ alpha**

Siehe [Liste, Objekte](#)

In der Eigenschaft befindet sich eine Zeichenkette mit der ODBC-Version, die der Treiber unterstützt. Die Version wird in der Form ##.## zurückgeliefert. Die ersten zwei Stellen kennzeichnen das Major-Release, die nächsten zwei Stellen das Minor-Release.

## Kontakt

---

### OdbcConDriverVersion

Versionsnummer des ODBC-Treibers

Typ [alpha](#)

Siehe [Liste, Objekte](#)

In dieser Eigenschaft steht die Versionsnummer und optional eine Beschreibung des Treibers. Die zurückgelieferte Zeichenkette hat die Form `##.##.####`. Die ersten zwei Stellen kennzeichnen das Major-Release, die nächsten zwei Stellen das Minor-Release und die letzten Stellen die Release-Version.

## Kontakt

OdbcConDSN



Name der Datenquelle

Typ alpha

Siehe Liste, Objekte

In dieser Eigenschaft steht der Name der Datenquelle, mit der das OdbcCon-Objekt verbunden ist.

## Kontakt

---

**OdbcConOdbcVersion**

**ODBC-Version der Bibliothek**

Typ [alpha](#)

Siehe [Liste, Objekte](#)

In der Eigenschaft wird ODBC-Version der Bibliothek (odbc32.dll) zurückgegeben.

## Kontakt

OdbcConReadOnly Schreibschutz   
der Datenquelle Typ logic

Siehe Liste, Objekte

Diese Eigenschaft zeigt an, ob die Datenquelle schreibgeschützt ist (true) oder nicht (false).

## Kontakt

**OdbcConServerName**



**Name des Servers der Datenquelle**

**Typ** alpha

**Siehe** Liste, Objekte

**In der Eigenschaft steht der Name des Servers der Datenquelle.**

## Kontakt

**OdbcConUserName**

**Name des Benutzers**

**Typ** alpha

**Siehe** Liste, Objekte

In dieser Eigenschaft befindet sich der bei der Anmeldung angegebene Benutzername (siehe OdbcConnect()). War keine Benutzeranmeldung erforderlich, ist die Eigenschaft leer.

## Kontakt

OdbcErrMsg ODBC-

Fehlerbeschreibung Typ alpha

Siehe Liste, Objekte

In dieser Eigenschaft befindet sich die Fehlerbeschreibung des zuletzt aufgetretenen ODBC-Fehlers.

Ist kein Fehler aufgetreten, ist die Zeichenkette leer.

## Kontakt

---

**OdbcErrSqlNativeCode**

Fehlerwert des ODBC-Treibers

Typ [int](#)

Siehe [Liste](#), [Objekte](#)

In dieser Eigenschaft steht der native Fehlerwert des ODBC-Treibers.

## Kontakt

OdbcErrSqlResult

SQL-Resultat

Typ int

Siehe Liste, Objekte

In dieser Eigenschaft steht der Resultatwert der zuletzt ausgeführten SQL-Funktion.

Ist kein Fehler aufgetreten, ist der Wert 0.

## Kontakt

OdbcErrSqlState Fehlerklasse und □

-unterklasse Typ alpha

Siehe Liste, Objekte

In dieser Zeichenkette befindet sich eine Zeichenkette in der in den ersten beiden Zeichen die Fehlerklasse und in den nächsten drei Zeichen die Fehlerunterklasse kodiert ist.

## Kontakt

OdbcResCountClm



Anzahl der Spalten in der Ergebnismenge

Typ [int](#)

Siehe [Liste](#), [Objekte](#)

In der Eigenschaft kann die Anzahl der Spalten in der Ergebnismenge abgefragt werden. Die Eigenschaft ist erst nach der Durchführung der [OdbcExecute\(\)](#)- oder [OdbcExecuteDirect\(\)](#)-Anweisung gesetzt. Die Eigenschaft wird nicht gesetzt, wenn keine Ergebnismenge ermittelt wurde.

## Kontakt

OdbcResCountRow



Anzahl der betroffenen Zeilen

Typ [int](#)

Siehe [Liste](#), [Objekte](#)

In der Eigenschaft kann die Anzahl der vom ODBC-Statement betroffenen Zeilen abgefragt werden.

Die Eigenschaft ist erst nach der Durchführung der [OdbcExecute\(\)](#)-oder [OdbcExecuteDirect\(\)](#)-

Anweisung gesetzt. Die Eigenschaft wird nicht gesetzt, wenn das Statement keine Zeilen verändert, eingefügt oder gelöscht hat.

## Kontakt

OdbcTblCatalog

**Katalog der Tabelle**

**Typ alpha**

**Siehe Liste, Objekte**

**In dieser Eigenschaft wird der Katalog der Tabelle ausgegeben. Der Katalog entspricht dem Namen der Datenbank.**

## Kontakt

OdbcTblName Name

der Tabelle Typ

alpha

Siehe Liste, Objekte

In der Eigenschaft wird der Name der Tabelle ausgegeben.

## Kontakt

OdbcTblRemarks



Beschreibung der Tabelle Typ

alpha

Siehe Liste, Objekte

In der Eigenschaft befindet sich die Beschreibung der Tabelle.

## Kontakt

OdbcTblSchema

Schema der Tabelle

Typ alpha

Siehe Liste, Objekte

In dieser Eigenschaft wird das Schema der verbundenen Datenquelle ausgegeben, sofern der entsprechende Treiber das unterstützt.

## Kontakt

**OdbcTblType**

**Typ der Tabelle**

**Typ** alpha

**Siehe** Liste,

**Siehe** Objekte

**In der Eigenschaft wird der Typ der Tabelle ausgegeben. In der Regel sind Tabellen vom Typ TABLE.**

## Kontakt

Options



Systemweite Optionen

Typ int  
Siehe Liste, Objekte,  
SysPropGet()

Über diese Eigenschaft werden systemweite Optionen festgelegt. Folgende Optionen können gesetzt werden:

ConvertUndef

Undefinierten Wert (-2.147.483.648) bei der Konvertierung mit CnvAI() wandeln.

DbaConnectOpErrorCode Fehlerwert bei Verbindungsabbruch zu verbundener

Datenbank bei Datensatzoperationen.  
Datensatzoperationen, die einen Deadlock (\_rDeadlock) erzeugen würden, führen statt dessen zum Laufzeitfehler ErrDeadLock.

## Kontakt

### Eigenschaften von System-Objekten - P bis Z

In dieser Liste sind alle Eigenschaften von P bis Z von System-Objekten aufgeführt. [Liste sortiert](#)

nach Gruppen,

Siehe [Alphabetische  
Liste aller  
Eigenschaften](#)

- [Parent](#)
- [PathAllAppData](#)
- [PathAllAutoStart](#)
- [PathAllDesktop](#)
- [PathAllPrograms](#)
- [PathAppData](#)
- [PathAppProg](#)
- [PathAppRoot](#)
- [PathAutoStart](#)
- [PathClient](#)
- [PathDesktop](#)
- [PathFonts](#)
- [PathMyDocuments](#)
- [PathMyMusic](#)
- [PathMyPictures](#)
- [PathMyVideo](#)
- [PathOffcAccess](#)
- [PathOffcExcel](#)
- [PathOffcOutlook](#)
- [PathOffcPowerPoint](#)
- [PathOffcWord](#)
- [PathPfmData](#)
- [PathPfmRoot](#)
- [PathProgramFiles](#)
- [PathProgramFilesCommon](#)
- [PathPrograms](#)
- [PathSystem](#)
- [PathTemp](#)
- [PathWindows](#)
- [PdfAttachCount](#)
- [PdfAuthor](#)
- [PdfCompany](#)
- [PdfCreator](#)
- [PdfKeywords](#)
- [PdfPageCount](#)
- [PdfPageHeight](#)
- [PdfPageRotation](#)
- [PdfPageWidth](#)
- [PdfPrintOrder](#)
- [PdfProducer](#)
- [PdfResolution](#)
- [PdfSubject](#)
- [PdfTitle](#)

- PdfUserRights
- PdfVersion
- PhysicalMemoryMB
- Platform
- PriorityIdle
- ProcCacheKB
- ProcCacheLimitKB
- ProcCurrent
- ProcCurrentFull
- ProcCurrentSub
- ProcessArchitecture
- ProcessExitCode
- ProcessID
- ProcessMemory
- ProcessMemoryKB
- ProcessMemoryLimitMB
- Protocol
- ProxyAuthorization
- ProxyAuthType
- Size
- SizeDba
- SizeDba64
- SizeOrg
- SizeOrg64
- StatusCode
- StopRequest
- StorageID
- SvcCallDelay
- SvcCallTime
- SvcDescription
- SvcName
- SvcSckHandle
- SvcSessionID
- TapiFlags
- TapiModuleFilename
- TapiNameComputer
- TapiOwnerRequest
- TapiUserName
- TerminalSession
- TerminalSessionID
- TimeExternal
- Type
- TypeMime
- TypeUser
- URI
- Validated
- ValidatedUser
- ValueAlpha
- ValueBigInt
- ValueCaltime
- ValueColor

## Kontakt

- ValueDate
- ValueDecimal
- ValueEvent
- ValueFloat
- ValueFont
- ValueInt
- ValueLogic
- ValuePoint
- ValueRange
- ValueRect
- ValueRtfTab
- ValueTime
- Version

## Kontakt

**Parent**

**Eltern-Objekt**

**Typ** handle

**Siehe** Liste,

**Siehe** Objekte

Diese Eigenschaft gibt den Deskriptor eines CteNode-Objekts zurück. Der zurückgegebene Knoten ist das Eltern-Objekt des angegebenen Objekts. Ist die Eigenschaft 0, ist kein Eltern-Objekt vorhanden.

Die Eigenschaft wird beim Einfügen des CteNode-Objekts mit CteInsertNode() oder CteInsert() in eine Struktur gesetzt. Sie kann nicht geschrieben werden.

## Kontakt

**PathAllAppData**  Allgemeines Verzeichnis der  
Applikationsdaten Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad der Programme für alle Benutzer im Startmenü ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathAllAppData // C:\ProgramData\Microsoft\Windows\Start Menu\Programs
```

## Kontakt

PathAllAutoStart



Allgemeines Verzeichnis für Autostart

Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad des Autostart-Verzeichnisses für alle Benutzer ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

Beispiel

```
_Sys->spPathAllAutoStart // C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
```

## Kontakt

---

### PathAllDesktop

Allgemeines Verzeichnis des Desktop

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Bei dem [Sys](#)-Objekt kann über diese Eigenschaft der Pfad des Desktop für alle Benutzer ermittelt werden.

**Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.**

### Beispiel

```
_Sys->spPathAllDesktop // C:\Users\Public\Desktop
```

## Kontakt

---

### PathAllPrograms

Allgemeines Verzeichnis für Programme

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Bei dem [Sys](#)-Objekt kann über diese Eigenschaft der Pfad der Applikationsdaten für alle Benutzer ermittelt werden.

**Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.**

### Beispiel

```
_Sys->spPathAllPrograms // C:\ProgramData
```

### PathAppData

Verzeichnis der Applikationsdaten

Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad der Applikationsdaten ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathAppData // C:\Users\admin\AppData\Roaming
```

## Kontakt

### PathAppProg

Verzeichnis des Clients

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad des Clients ermittelt werden. Jeder Client ermittelt dabei seinen eigenen Pfad, da unterschiedliche Clients in unterschiedliche Pfade installiert werden.



Der zurückgegebene Pfad entspricht nicht dem Pfad, über den der Client gestartet wurde, sondern immer dem Verzeichnis der Installation.

### Beispiel

```
_Sys->spPathAppProg // C:\Program Files\CONZEPT 16\Client
```

### PathAppRoot

**CONZEPT 16 Installationsverzeichnis**

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad von CONZEPT 16 ermittelt werden. Bei der Standard-Installation wird der Pfad 'C:\Program Files\CONZEPT 16' zurückgegeben.



Der zurückgegebene Pfad entspricht nicht dem Pfad, über den der Client gestartet wurde, sondern immer dem Verzeichnis der Installation.

### Beispiel

```
_Sys->spPathAppRoot // C:\Program Files\CONZEPT 16
```

## Kontakt

PathAutoStart Verzeichnis

für Autostart Typ

alpha(8192)

Siehe Liste, Objekte,  
SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad des Autostart-Verzeichnisses ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

Beispiel

```
_Sys->spPathAutoStart // C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Sta
```

## Kontakt

**PathClient**



Pfad, aus dem der Client gestartet wurde

Typ [alpha\(8192\)](#)

Siehe [Liste, Objekte, SysPropGet\(\)](#)

Bei dem [Sys](#)-Objekt kann über diese Eigenschaft der Pfad, aus dem der Client gestartet wurde, ermittelt werden.

**Beispiel**

```
_Sys->spPathClient // C:\Program Files\CONZEPT 16\Client\
```

## Kontakt

---

**PathDesktop**

**Verzeichnis des Desktop**

**Typ** alpha(8192)

**Siehe** Liste, Objekte,

**Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad des Desktop ermittelt werden.**

**Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.**

**Beispiel**

```
_Sys->spPathDesktop // C:\Users\admin\Desktop
```

### PathFonts

Verzeichnis für die Schriftarten

Typ [alpha\(8192\)](#)

Siehe [Liste, Objekte](#),

Bei dem [Sys](#)-Objekt kann über diese Eigenschaft das Verzeichnis der Schriftarten ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathFonts // C:\Windows\Fonts
```

### PathMyDocuments

Verzeichnis "Eigene Dokumente"

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Bei dem [Sys](#)-Objekt kann über diese Eigenschaft der Pfad für die eigenen Dokumente ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathMyDocuments // C:\Users\admin\Documents
```

## Kontakt

---

### PathMyMusic

Verzeichnis für eigene Musik

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad für die eigene Musik ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathMyMusic // C:\Users\admin\Music
```

## Kontakt

---

### PathMyPictures

Verzeichnis für eigene Bilder

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad für die eigenen Bilder ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathMyPictures // C:\Users\admin\Pictures
```

## Kontakt

---

### PathMyVideo

Verzeichnis für eigene Videos

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad für eigene Videos ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathMyVideo // C:\Users\admin\Videos
```

## Kontakt

**PathOffcAccess**  Installationsverzeichnis von  
Microsoft Access Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad vom Microsoft Access ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathOffcAccess // C:\Program Files (x86)\Microsoft Office\Office14\
```

## Kontakt

**PathOffcExcel**  Installationsverzeichnis von  
Microsoft Excel Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad vom Microsoft Excel ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

**Beispiel**

```
_Sys->spPathOffcExcel // C:\Program Files (x86)\Microsoft Office\Office14\
```

## Kontakt

PathOffcOutlook ↴ Installationsverzeichnis von

Microsoft Outlook Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad vom Microsoft Outlook ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

**Beispiel**

```
_Sys->spPathOffcOutlook // C:\Program Files (x86)\Microsoft Office\Office14\
```

## Kontakt

**PathOffcPowerPoint** Installationsverzeichnis von Microsoft  
PowerPoint Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad vom Microsoft PowerPoint ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

**Beispiel**

```
_Sys->spPathOffcPowerPoint // C:\Program Files (x86)\Microsoft Office\Office14\
```

## Kontakt

PathOffcWord | Installationsverzeichnis von  
Microsoft Word Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Installationspfad vom Microsoft Word ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathOffcWord // C:\Program Files (x86)\Microsoft Office\Office14\
```

## Kontakt

**PathPfmData**  Pfad von Client-Konfigurationsdateien der CONZEPT 16-Installation unter Windows  
**Typ alpha(8192)**

Siehe [Liste, Objekte, SysPropGet\(\)](#)

Bei dem [Sys-Objekt](#) kann über diese Eigenschaft der Pfad der Client-Konfigurationsdateien der CONZEPT 16-Installation unter Windows ermittelt werden. Dieser Pfad ist abhängig vom verwendeten Client und liefert daher auch unter dem gleichen Betriebssystem verschiedene Ergebnisse.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

**Beispiel:**

```
tPathPfmData # _Sys->spPathPfmData;
```

Für tPathPfmData ergibt sich unter Windows 7 beispielsweise folgender Inhalt:

*Ausgeführt mit dem Standard-Client*

```
'C:\ProgramData\CONZEPT 16\Client'
```

*Ausgeführt mit dem Druckprozessor*

```
'C:\ProgramData\CONZEPT 16\Printer'
```

*Ausgeführt per Remote-Prozedur auf dem Server*

```
'C:\ProgramData\CONZEPT 16\Server'
```



Für die Standardverzeichnisse von Konfigurationsdateien unter Windows siehe auch [Speicherorte von Konfigurationsdateien](#).



### PathPfmRoot

Pfad von Konfigurationsdateien der CONZEPT 16-Installation unter Windows Typ  
alpha(8192)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Bei dem Objekt Sys kann über diese Eigenschaft der Root-Pfad der Konfigurationsdateien der CONZEPT 16-Installation unter Windows ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

Beispiel:

```
tPathPfmRoot # _Sys->spPathPfmRoot;
```

Für tPathPfmRoot ergibt sich unter Windows 7 beispielsweise folgender Inhalt:

```
'C:\ProgramData\CONZEPT 16'
```



Für die Standardverzeichnisse von Konfigurationsdateien unter Windows siehe auch  
[Speicherorte von Konfigurationsdateien](#).

### PathProgramFiles

Verzeichnis der Programmdateien

Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad ermittelt werden, in dem die Programme installiert werden.

### Beispiel

```
_Sys->spPathProgramFiles // C:\Program Files (x86)
```

## Kontakt

**PathProgramFilesCommon**    Pfad der gemeinsam genutzten  
Programmdateien Typ alpha(8192)

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad ermittelt werden, in dem gemeinsame Daten von Programmen abgelegt werden.

**Beispiel**

```
_Sys->spPathProgramFilesCommon // C:\Program Files (x86)\CommonFiles
```

## Kontakt

---

### PathPrograms

Verzeichnis der Programme

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad der Programme im Startmenü ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

### Beispiel

```
_Sys->spPathPrograms // C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
```

### PathSystem

#### Windows-Systemverzeichnis

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Systempfad des Betriebssystems ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

#### Beispiel

```
_Sys->spPathSystem // C:\Windows\system32
```

### PathTemp

Pfad für temporäre Dateien

Typ alpha(8192)

Siehe Liste, Objekte,

Bei dem Sys-Objekt kann über diese Eigenschaft der Pfad für temporäre Dateien, wie in der Betriebssystemumgebung definiert, ermittelt werden.

Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.

Beispiel:

```
WinDialogBox(0, 'Pfad für temp. Dateien', _Sys->spPathTemp, _WinIcoInformation, _Win
```

## Kontakt

---

**PathWindows**

**Pfad des Betriebssystems**

**Typ** alpha(8192)

**Siehe** Liste, Objekte,

**Bei dem Sys-Objekt kann über diese Eigenschaft der Root-Pfad des Betriebssystems ermittelt werden.**

**Der Pfad muss nicht bei allen Betriebssystemen gesetzt sein. Existiert der entsprechende Eintrag nicht, liefert die Eigenschaft einen Leerstring zurück.**

**Beispiel**

```
_Sys->spPathWindows // C:\Windows
```

## Kontakt

**PdfAttachCount**  Anzahl der Anhänge eines PDF-Dokumentes Typ int

Siehe Liste, Objekte

Mit der Eigenschaft kann die Anzahl der Anhänge eines PDF-Dokuments ermittelt werden.

**Beispiel:**

```
// Anhänge zähltentAttachCount # tHdlPdf->spPdfAttachCount;// Anhänge durchgehenfor tAttachNo #
```

## Kontakt

---

### PdfAuthor

Verfasser des PDF-Dokumentes

Typ [alpha\(8192\)](#)

Siehe [Liste](#), [Objekte](#), [Beispiel](#)

Über diese Eigenschaft wird bei der Erzeugung eines PDF-Dokuments der Verfasser des Dokumentes gesetzt oder ermittelt (Dokumentzusammenfassung "Verfasser").

## Kontakt

PdfCompany



Firma des PDF-Dokuments

Typ alpha(8192)

Siehe Liste, Objekte

Mit dieser Eigenschaft kann die Firma, die das PDF-Dokument erstellt hat, abgefragt oder gesetzt werden. Dies ist eine benutzerdefinierte Eigenschaft.

## Kontakt

---

**PdfCreator**

**Ersteller des PDF-Dokumentes**

**Typ alpha(8192)**

**Siehe Liste, Objekte, Beispiel**

**Über diese Eigenschaft wird der Ersteller des Dokumentes**

**(Dokumentzusammenfassung "Erstellt mit") gesetzt oder ermittelt.**

## Kontakt

---

### PdfKeywords

Schlüsselwörter des PDF-Dokuments

Typ [alpha\(8192\)](#)

Siehe [Liste, Objekte](#)

In dieser Eigenschaft werden die Stichwörter des PDF-Dokuments eingetragen. Die Eigenschaft kann nur gelesen werden.

## Kontakt

PdfPageCount  Anzahl der Seiten in einem PDF-

Dokument Typ int

Siehe Liste, Objekte

Über diese Eigenschaft kann die Anzahl der Seiten in einem PDF-Dokument ermittelt werden.

**PdfPageHeight**

Seitenhöhe eines PDF-Dokuments

Typ **float**

Siehe [Liste](#), [Objekte](#), [PdfPageWidth](#)

Mit dieser Eigenschaft kann die Höhe einer Seite eines PDF-Dokuments ermittelt und gesetzt werden.

Die Höhe ist in Millimetern angegeben.

### PdfPageRotation

Rotation der geöffneten Seite

Typ [int](#)

Siehe [Liste, Objekte](#)

Mit dieser Eigenschaft kann die mit [PdfPageOpen\(\)](#) geöffnete Seite gedreht werden. Die Seite kann in Winkeln von Vielfachen von 90 Grad gedreht werden. Werte größer 0 entsprechen einer Rotation im Uhrzeigersinn, Werte kleiner 0 gegen den Uhrzeigersinn.

Die Rotation wirkt sich nicht auf die Seitenbreite und Seitenhöhe aus. Die Seite wird erst bei der Anzeige (zum Beispiel durch einen PDF-Reader) gedreht. Nachfolgende Inhalte sind von der Rotation nicht betroffen.

Ist zum Ausführungszeitpunkt der Anweisung keine Seite geöffnet, wird der Laufzeitfehler [ErrPropInvalid](#) erzeugt. Bei Angaben, die keine Vielfachen von 90 Grad sind, erfolgt der Laufzeitfehler [ErrValueInvalid](#).

## Kontakt

---

### PdfPageWidth

Breite der Seite des PDF-Dokuments

Typ `float`

Siehe [Liste](#), [Objekte](#), [PdfPageHeight](#)

Mit dieser Eigenschaft kann die Seitenbreite eines PDF-Dokuments ermittelt und gesetzt werden. Die Breite ist in Millimetern angegeben.

## Kontakt

---

**PdfProducer**

**Ersteller des Pdf-Dokuments**

**Typ alpha(8192)**

**Siehe Liste, Objekte**

**Mit dieser Eigenschaft kann der Hersteller des PDF-Dokuments ermittelt und gesetzt werden.**

**Ist die Eigenschaft leer, wird sie beim Speichern der PDF-Datei auf 'CONZEPT 16 ' und die komplette Versionsnummer gesetzt.**

## Kontakt

**PdfResolution**



**Auflösung des PDF-Dokuments**

**Typ** `float`

**Siehe** [Liste, Objekte](#)

Mit dieser Eigenschaft kann die Auflösung des PDF-Dokuments ermittelt werden. Die horizontale und vertikale Auflösung ist identisch. Der Wert wird in DPI zurückgegeben.

## Kontakt

PdfSubject



Thema des PDF-Dokuments

Typ alpha(8192)

Siehe Liste, Objekte

Über diese Eigenschaft kann das Thema eines PDF-Dokuments  
(Dokument-Information Thema) ermittelt und gesetzt werden.

## Kontakt

PdfTitle



**Titel des PDF-Dokumentes**

Typ alpha(8192)

Siehe Liste, Objekte, Beispiel

Über diese Eigenschaft kann der Titel eines PDF-Dokuments  
(Dokumentzusammenfassung "Titel") gesetzt oder abgefragt werden.

### PdfUserRights

#### Berechtigungen in dem Dokument

Typ `int`

Siehe [Liste, Objekte](#)

In dieser Eigenschaft werden die Berechtigungen eines PDF-Dokuments angegeben.

Zum Setzen der Berechtigungen können folgende Konstanten verwendet werden:

- `_PdfRightDenyNothing`

Es werden keine Rechte eingeschränkt (default).

- `_PdfRightDenyAll`

Alle Einschränkungen werden gesetzt.

- `_PdfRightDenyPrint`

Das Drucken ist nicht erlaubt.

- `_PdfRightDenyModify`

Das Ändern von Inhalten ist nicht erlaubt.

- `_PdfRightDenyCopy`

Das Kopieren von Inhalten ist nicht erlaubt.

- `_PdfRightDenyExtract`

Das Extrahieren von Inhalten ist nicht erlaubt.

Die Konstanten können miteinander kombiniert werden.

Beispiele:

```
tPdf->spPdfUserRights # _PdfRightDenyCopy | _PdfRightDenyModify; // Kopieren und Ändern verboten
```

## Kontakt

---

### PdfVersion

**PDF Version des PDF-Dokuments**

Typ int

Siehe Liste, Objekte

Mit der Eigenschaft kann die PDF-Version eines PDF-Dokuments ermittelt werden. Die ermittelte Zahl entspricht der zweiten Nummer der Version.

**Beispiel:**

```
tVersion # 'PDF Version 1.' + CnvAI(tHdlPdf->spPdfVersion);
```

## Kontakt

---

### PhysicalMemoryMB

Größe des verfügbaren Arbeitsspeichers

Typ int

Siehe Liste, Objekte, SysPropGet()

Über diese Eigenschaft kann die Größe des physikalisch zur Verfügung stehenden Arbeitsspeichers des Rechners, auf dem der Client läuft, in MB ermittelt werden.

Beispiel:

```
tMemory # _Sys->spPhysicalMemoryMB;
```

## Kontakt

Platform

Prozedurumgebung

Typ int

Siehe Liste, Objekte,  
SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft die Prozedurumgebung ermittelt werden.

Folgende Werte kann die Eigenschaft annehmen:

PfmAPI Die Prozedur wird über die Programmierschnittstelle ausgeführt

PfmClient3 Die Prozedur wird auf dem Client für Datenbanken der Version 3.x ausgeführt

PfmClient4 Die Prozedur wird auf dem Standard-Client ausgeführt

PfmClient5 Die Prozedur wird auf dem Advanced-Client ausgeführt

PfmODBC Die Prozedur wird über die ODBC-Schnittstelle ausgeführt

PfmPHP Die Prozedur wird über die PHP-Schnittstelle ausgeführt

PfmPrinter Die Prozedur wird über den Druckprozessor ausgeführt

PfmServer Die Prozedur wird auf dem Datenbank-Server ausgeführt

PfmWeb Die Prozedur wird über die Web-Schnittstelle ausgeführt

PfmSOA Die Prozedur wird über den SOA-Service ausgeführt

### PriorityIdle

Prozess-Priorität verändern

Typ [logic](#)

Siehe [Liste, Objekte](#)

Bei dem [Sys-Objekt](#) kann die Prozess-Priorität verändert werden.

Ist diese Eigenschaft auf [true](#) gesetzt, wird der Client-Prozess vom Betriebssystem mit einer niedrigeren Priorität behandelt, d. h. er bekommt weniger Prozessorzeit zugeteilt. Diese Eigenschaft sollte nur mit Jobservern eingesetzt werden, um die Parallelität auf dem System zu gewährleisten.

**Beispiel:**

```
_Sys->spPriorityIdle # true;
```

### ProcCacheKB

Belegung des Prozedurcaches

Typ int

Liste, Objekte,

Siehe ProcCacheLimitKB,

SysPropGet()

Über diese Eigenschaft kann die aktuelle Belegung des Prozedurcaches ermittelt werden. Die Speichermenge ist in KB angegeben. Die Eigenschaft kann mit ProcCacheLimitKB verglichen werden.

Beispiel:

```
tCacheUse # _Sys->spProcCacheKB;
```

**ProcCacheLimitKB**

**Maximalgröße des Prozedurcaches**

**Typ** int

**Siehe** Liste, Objekte, ProcCacheKB

Über diese Eigenschaft kann die Maximalgröße des Prozedurcaches ermittelt und gesetzt werden. Die Speichermenge ist in KB angegeben. Die Maximalgröße kann zwischen 256 und 65536 (64 MB) angegeben werden.

**Beispiel:**

```
tCacheUse # _Sys->spProcCacheKB;
```

ProcCurrent

### **Aktuelle Prozedur ermitteln**

Typ alpha(20)

Siehe Liste, Objekte,

## Siehe

 Diese Systemeigenschaft wurde durch PROC abgelöst und sollte nicht mehr verwendet werden.

Bei dem Objekt **System** kann die aktuell ausgeführte Prozedur ermittelt werden. Das Setzen der Eigenschaft ist bei diesem Objekt nicht möglich.

Die Eigenschaft **ProcCurrentFull** liefert den Namen der Prozedur und der Funktion zurück.

## **Beispiel:**

```
WinDialogBox(0, 'Verarbeitungsfehler', 'In der Prozedur ' + Sys->spProcCurrent + '
```

**i** Durch das Deaktivieren der internen Aufrufprotokollierung von Prozeduren und Funktionen über den Befehl DbgControl() mit dem Parameter DbgOff entfällt die Möglichkeit aktuelle Prozedurinformationen aus den Systemeigenschaften auszulesen. Mit den Compilermacros ist dies weiterhin möglich.

## Kontakt

**ProcCurrentFull**  Ermitteln der aktuellen Prozedur und  
Funktion Typ alpha(101)

Siehe Liste, Objekte, SysPropGet()



Diese Systemeigenschaft wurde durch PROCFUNC abgelöst und sollte nicht mehr verwendet werden.

Bei dem Objekt System kann die aktuell ausgeführte Prozedur und Funktion ermittelt werden. Das Setzen der Eigenschaft ist bei diesem Objekt nicht möglich.

Als Ergebnis wird die aktuelle Prozedur (maximal 20 Zeichen) und die aktuelle Funktion (maximal 80 Zeichen), durch einen Doppelpunkt getrennt, zurückgegeben.

Beispiel:

```
WinDialogBox(0, 'Verarbeitungsfehler', 'In der Prozedur ' + _Sys->spProcCurrentFull +
```



Durch das Deaktivieren der internen Aufrufprotokollierung von Prozeduren und Funktionen über den Befehl DbgControl() mit dem Parameter DbgOff entfällt die Möglichkeit aktuelle Prozedurinformationen aus den Systemeigenschaften auszulesen. Mit den Compilermacros ist dies weiterhin möglich.

## ProcCurrentSub

## Ermitteln der aktuellen Funktion

Typ alpha(80)

**Siehe Liste, Objekte, SysPropGet()**

**i** Diese Systemeigenschaft wurde durch FUNC abgelöst und sollte nicht mehr verwendet werden.

Bei dem Objekt System kann die aktuell ausgeführte Funktion ermittelt werden. Das Setzen der Eigenschaft ist bei diesem Objekt nicht möglich.

**Die Eigenschaft ProcCurrentFull liefert den Namen der Prozedur und der Funktion zurück.**

## Beispiel:

```
WinDialogBox(0, 'Verarbeitungsfehler', 'In der Funktion ' + Sys->spProcCurrentSub +
```

**i** Durch das Deaktivieren der internen Aufrufprotokollierung von Prozeduren und Funktionen über den Befehl [DbgControl\(\)](#) mit dem Parameter [DbgOff](#) entfällt die Möglichkeit aktuelle Prozedurinformationen aus den Systemeigenschaften auszulesen. Mit den [Compilermacros](#) ist dies weiterhin möglich.

## Kontakt

ProcessArchitecture  Prozess-Architektur der CONZEPT 

**16-Anwendung Typ int**

Siehe Liste, Objekte, SysPropGet()

Über diese Eigenschaft kann die Architektur der CONZEPT **16-Anwendung** (32- bzw. 64-Bit) ermittelt werden.

Beispiel:

```
tIs64Bit # _Sys->spProcessArchitecture = 64;
```

## Kontakt

ProcessExitCode



Externer Rückgabewert der Applikation

Typ byte

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft für den Standard- und Advanced-Client der externe Rückgabewert der Applikation gesetzt werden. Dieser kann dann von externen Programmen nach dem Beenden des Clients ermittelt werden.

## Kontakt

**ProcessID**

Prozess-ID ermitteln

Typ int  
Siehe Liste, Objekte,

Bei dem Objekt System kann die Prozess-ID (PID) des CONZEPT 16-Clients ermittelt werden. Wird die Anweisung auf dem Server ausgeführt (RmtCall()), wird die Prozess-ID des Datenbankprozesses zurückgegeben.

Die Prozess-ID kann zum Beispiel verwendet werden, um eindeutige Namen für temporäre Dateien zu erzeugen.

## Kontakt

ProcessMemory  Speicherverbrauch der CONZEPT 16-

Anwendung Typ int

Siehe Liste, Objekte, ProcessMemoryKB

Über diese Eigenschaft wird der Speicher ermittelt, den die CONZEPT 16-Anwendung verbraucht (beispielsweise durch geladene Dialoge, globale Variablenbereiche usw.). Die Speichermenge ist in Bytes angegeben.

Beispiel:

```
tMemUsage # _Sys->spProcessMemory;
```

## Kontakt

**ProcessMemoryKB**    Speicherverbrauch der CONZEPT 16-  
Anwendung Typ int

Siehe Liste, Objekte, SysPropGet()

Über diese Eigenschaft wird der Speicher ermittelt, den die CONZEPT 16-Anwendung verbraucht (beispielsweise durch geladene Dialoge, globale Variablenbereiche usw.). Die Speichermenge wird in KB angegeben. Wird eine genauere Angabe benötigt, kann die Eigenschaft ProcessMemory verwendet werden.

Beispiel:

```
tMemUsage # _Sys->spProcessMemoryKB;
```

## Kontakt

ProcessMemoryLimitMB  $\sqsubset$  Speicherreservierung des  
CONZEPT 16-Clients Typ int

Siehe Liste, Objekte, SysPropGet()

Über diese Eigenschaft wird der Speicher ermittelt, den der CONZEPT 16-Client reservieren kann.

Beispiel:

```
tMemReserved # _Sys->spProcessMemoryLimitMB * 1024; // Value in KB
```

### Protocol

#### Protokoll der HTTP-Anfrage

Typ alpha(15)

Liste, Objekte,

Siehe SysPropGet(),

SysPropSet()

Eigenschaft des HTTP-Objekts. Die Eigenschaft enthält das Protokoll und die Version des Protokolls.

Die Eigenschaft kann bei den Objekt-Typen HttpRecvRequest und HttpRecvResponse nur gelesen werden.

Bei den Typen HttpSendRequest und HttpSendResponse ist die Eigenschaft mit 'HTTP/1.1' vorbelegt. Beim Empfang (HttpRecv...) werden nur die HTTP-Versionen 1.0 und 1.1 unterstützt.

### Beispiel:

```
switch (tRequest->spProtocol){ case 'HTTP/1.1' : ... case 'HTTP/1.0' : ... }
```

### ProxyAuthorization

Autorisierung beim HTTP-Proxy

Typ alpha(63)

Siehe Liste, Objekte,

Siehe SysPropGet(), SysPropSet()

Eigenschaft des HTTP-Objekts. Die Eigenschaft steht nur beim Objekt-Typ HttpSendRequest zur Verfügung und wird nur verwendet, wenn die Basic-Authentifizierung aktiviert ist, ansonsten wird sie ignoriert (siehe Eigenschaft ProxyAuthType).

**Beispiel:**

```
tSendRequest->spProxyAuthType # _HttpProxyAuthTypeBasic;tSendRequest->spProxyAuthorization # $Pro
```

## Kontakt

---

### ProxyAuthType

Autorisierungsart beim HTTP-Proxy

Typ alpha(63)

Siehe Liste, Objekte, SysPropGet(),

Eigenschaft des HTTP-Objekts. Die Eigenschaft definiert die Art der Authorisierung bei Verbindung mit einem Proxy-Server.

HttpProxyAuthTypeBasic Basic-Autorisierung. Bei HttpClose() muss die Konstante HttpUseWebProxy angegeben werden. Benutzername und Kennwort werden über die Eigenschaft ProxyAuthorization definiert. Die Eigenschaft muss im Format <Benutzer>:<Kennwort> definiert werden.

HttpProxyAuthTypeNTLM Proxy-Autorisierung via NTLM. Bei SckClose muss die Konstante HttpUseWebProxy angegeben werden. Der Inhalt der Eigenschaft ProxyAuthorization wird ignoriert.

Beispiel:

```
tSendRequest->spProxyAuthType # _HttpProxyAuthTypeBasic;tSendRequest->spProxyAuthorization # $Pro
```

## Kontakt

**Objekte von ProxyAuthType**

Folgende Objekte unterstützen die Eigenschaft ProxyAuthType

ProxyAuthType,

Liste der

Siehe Objekte, Liste

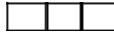
der

Eigenschaften

- HTTP

## Kontakt

Size



Größe / Breite des Objektes

Typ int

Siehe Liste, Objekte

Je nach Objekt hat diese Eigenschaft folgende Bedeutung:

- Axis

Mit dieser Eigenschaft wird die horizontale bzw. vertikale Ausdehnung der Achse (Titel + Skala) festgelegt.

- PrtGanttAxis

Mit dieser Eigenschaft wird die horizontale bzw. vertikale Ausdehnung der Achse (Titel + Skala) festgelegt.

- Toolbar und Statusbar

Höhe (bei Position am oberen oder unteren Rand) bzw. die Breite (bei Position am linken oder rechten Rand) des Toolbar- und Statusbar-Objekts.

Die Höhe bzw. die Breite der Toolbar/Statusbar verändert sich erst, wenn die durch die Schaltflächen vorgegebene Mindesthöhe überschritten wird. Wird ein Wert kleiner 0 angegeben, entspricht die Höhe bzw. die Breite immer dem Absolutwert der Eigenschaft. Die Schaltflächen werden innerhalb der Toolbar bzw. Statusbar zentriert dargestellt, d. h. der obere und untere Rand bzw. der linke und rechte Rand werden gleichmäßig abgeschnitten.

- Toolbar-Button und Statusbar-Button Breite der Schaltfläche.

- Memory

Dies ist die Größe des Speicherbereichs in Bytes, wie sie in MemAllocate() angegeben wurde. Bei einigen Befehlen kann bei der Datenlänge die Konstante MemObjSize übergeben werden, wodurch dann dieser Wert verwendet wird. Die Eigenschaft kann bei diesem Objekt nur gelesen werden.

## Kontakt

**SizeDba**  Speicherverbrauch des Objekts in der Datenbank

Typ int

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#), [SizeDba64](#)

Diese Eigenschaft enthält die Größe in Bytes, die der Objektinhalt in der Datenbank belegt. Die Größe ist immer ein Mehrfaches von 2 KB. Der Wert ist 0, wenn das Objekt leer ist. Die Eigenschaft kann nur gelesen werden.

 Mit dieser Eigenschaft kann nur die Größe von [Binäre Objekten](#), die kleiner als 2 GB sind, korrekt ermittelt werden. Für größere [Binäre Objekte](#) muss [SizeDba64](#) verwendet werden.

Ist bei einem [binären Objekt](#) die Größe von 2 GB überschritten, liefert diese Eigenschaft [MaxInt](#) zurück.

## Kontakt

### SizeDba64



Speicherverbrauch des Objekts in der Datenbank (als 64-Bit-Wert)

Typ [bigint](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#), [SizeDba](#)

Diese Eigenschaft enthält die Größe in Bytes, die der Objektinhalt in der Datenbank belegt. Die Größe ist immer ein Mehrfaches von 2 KB. Der Wert ist 0, wenn das Objekt leer ist. Die Eigenschaft kann nur gelesen werden.



Im Gegensatz zu [SizeDba](#) kann mit dieser Eigenschaft auch die Größe von [binären Objekten](#), die größer als 2 GB sind, korrekt ermittelt werden. Die maximale Größe, die korrekt ermittelt werden kann, ist 2 TB.

### SizeOrg

Originalgröße des Objekts

Typ int

Liste, Objekte,

Siehe SysPropGet(),

SizeOrg64

Diese Eigenschaft enthält die Originalgröße des Objektinhalts in Bytes. Die Eigenschaft kann nur gelesen werden.



Mit dieser Eigenschaft kann nur die Größe von Binäre Objekten, die kleiner als 2 GB sind, korrekt ermittelt werden. Für größere Binäre Objekte muss SizeOrg64 verwendet werden.

Ist bei einem binären Objekt die Größe von 2 GB überschritten, liefert diese Eigenschaft MaxInt zurück.

## Kontakt

**SizeOrg64**  Originalgröße des Objekts (als 64-Bit-Wert) Typ bigint

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#), [SizeOrg](#)

Diese Eigenschaft enthält die Originalgröße des Objektinhalts in Bytes. Die Eigenschaft kann nur gelesen werden.



Im Gegensatz zu [SizeOrg](#) kann mit dieser Eigenschaft auch die Größe von binären Objekten, die größer als 2 GB sind, korrekt ermittelt werden. Die maximale Größe, die korrekt ermittelt werden kann, ist 2 TB.

## Kontakt

---

### StatusCode

Status einer HTTP-Kommunikation

Typ [alpha\(63\)](#)

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#),

Eigenschaft des [HTTP](#)-Objekts. In dieser Eigenschaft wird der Status der HTTP-Kommunikation abgelegt. Die Eigenschaft steht nur bei [HTTP](#)-Objekten vom Typ [HttpSendResponse](#) und [HttpRecvResponse](#) zur Verfügung. Beim Typ [HttpRecvResponse](#) kann die Eigenschaft nur gelesen werden.

Der Statuscode wird in Form eines 3-stelligen Statuswerts und einem nachfolgenden Statustext angegeben. Beim Typ [HttpSendResponse](#) ist die Eigenschaft mit '200 OK' vorbelegt.

## Kontakt

---

### StopRequest

Beenden der Verarbeitung angefordert

Typ logic

Siehe Liste, Objekte, SysPropGet()

Diese Eigenschaft ist true, wenn der Task gestoppt werden soll. Innerhalb einer Jobprozedur ist die Eigenschaft true, wenn der Task oder der Job gestoppt werden soll. Die Eigenschaft sollte bei Verarbeitungen, die länger als 10 Sekunden dauern, in regelmäßigen Abständen überprüft werden. Falls die Eigenschaft true ist, sollte die Prozedur möglichst schnell beendet werden.

Die Eigenschaft wird zum Beispiel dann gesetzt, wenn der Client, der SOA-Service oder der Computer angehalten werden soll.

Die Eigenschaft kann nur gelesen werden.

## Kontakt

StorageID



Interne (eindeutige) ID  
des Objekts

Typ **int**

Siehe [Liste](#), [Objekte](#), [SysPropGet\(\)](#),  
[SysPropSet\(\)](#)

In dieser Eigenschaft steht die interne ID des binären Objekts. Diese Eigenschaft kann nur gelesen werden.

### SvcCallDelay

Zeit bis zum nächsten Aufruf

Typ int

Liste, Objekte,

Siehe SvcCallTime,

SysPropGet()

Dies ist die aktuell verwendete Zeitspanne in Sekunden, die nach dem Ende der Ereignisprozedur vergeht, bevor ein neues Ereignis ausgelöst wird. Das Setzen der Zeitspanne ist wirkungslos, wenn ebenfalls die Eigenschaft SvcCallTime gesetzt wird.

Wird weder diese Eigenschaft noch die Eigenschaft SvcCallTime gesetzt, wird der Task nach der Zeit in der Einstellung time\_delay\_default erneut aufgerufen.



Ist der nächste Ausführungszeitpunkt mehr als 24 Stunden in der Zukunft, wird das Ereignis in 24 Stunden erneut ausgeführt.

### SvcCallTime

Zeitpunkt für den nächsten Aufruf

Typ [bigint](#)

[Liste](#), [Objekte](#), [SvcCallDelay](#),

Siehe

Dies ist der Zeitpunkt, zu dem ein neues Ereignis ausgelöst wird. Der Zeitpunkt wird als Zeitstempel im UTC-Format in die Eigenschaft geschrieben. Eine Variable vom Typ [caltime](#) kann mit der Anweisung [CnvBC\(\)](#) entsprechend konvertiert werden.



Durch die Angabe eines Zeitstempels im UTC-Format kann sich der Zeitpunkt bei Sommer-/Winterzeitumstellung verschieben, wenn eine [caltime](#)-Variable mit [CnvBC\(\)](#) konvertiert wird.

Wird weder diese Eigenschaft noch die Eigenschaft [SvcCallDelay](#) gesetzt, wird der [Task](#) nach der Zeit in der Einstellung [time delay default](#) erneut aufgerufen.



Ist der nächste Ausführungszeitpunkt mehr als 24 Stunden in der Zukunft, wird das Ereignis in 24 Stunden erneut ausgeführt.

## Kontakt

---

### SvcDescription

**Beschreibung des Task oder des Jobs**

**Typ** [alpha\(8192\)](#)

**Siehe** [Liste](#), [Objekte](#), [SysPropGet\(\)](#)

Dies ist die Beschreibung des Tasks. Die Beschreibung ist identisch für alle Ereignisse, kann jedoch für jeden gestarteten **Job** abweichend definiert werden (siehe [JobStart\(\)](#)).

**Die Eigenschaft kann nur gelesen werden.**

## Kontakt

---

### SvcName

Name des Task oder Jobs

Typ alpha(128)

Siehe Liste, Objekte,

Dies ist der Name des Tasks. Der Name ist identisch für alle Ereignisse und für alle Jobs, die innerhalb des Tasks gestartet werden. Der Name entspricht der Eintragung [Name] in der Konfigurationsdatei des SOA-Service. Er kann maximal 128 Zeichen lang sein.

Die Eigenschaft kann nur gelesen werden.

### SvcSckHandle

Deskriptor der Socket-Verbindung

Typ handle

Siehe Liste, Objekte, SysPropGet()

Diese Eigenschaft kann nur dann gelesen werden, wenn ein Task mit der Betriebsart SOCKET gestartet wurde. Die Eigenschaft enthält den Deskriptor der Socket-Verbindung, der für den Datenaustausch mit der Gegenstelle verwendet wird.

Die Eigenschaft kann nur gelesen werden.

## Kontakt

**SvcSessionID**

**Session-Id des Task**

Typ int

Siehe Liste, Objekte

In dieser Eigenschaft wird die Session-Id des Task abgelegt. Die Session-Id wird mit der Anweisung SvcSessionControl() erzeugt.

### TapiFlags

Funktionen des TAPI-Gerätes

Typ int

Siehe Objekte

Über die Eigenschaft kann das Verhalten eines TapiDevice gesteuert werden.

Folgende Ausprägung steht zur Verfügung:

- TapiListenMonitor

Ist diese Ausprägung gesetzt, werden beim nächsten Aufruf von TapiListen() keine Besitzrechte an dem Gerät angefordert. Dadurch können bestimmte Funktionen nicht durchgeführt werden (siehe TAPI-Befehle). Es ist aber eine Überwachung (Monitoring) des Gerätes möglich.

- TapiFlagsDefault

Mit dieser Konstanten können der voreingestellte Wert wieder hergestellt werden.

## Kontakt

**TapiModuleFilename**

**Name des Moduls**

**Typ** alpha(256)

**Siehe** Liste, Objekte

In dieser Eigenschaft steht der Name des Moduls, das ein TAPI-Gerät verwendet. Der Name der Applikation wird in der Eigenschaft Name abgelegt. Wird das Gerät von einer CONZEPT 16-Applikation überwacht, steht in dieser Eigenschaft 'c16\_objw.dll'.

## Kontakt

TapiNameComputer

Rechnername

Typ [alpha\(256\)](#)

Siehe [Liste, Objekte](#)

In dieser Eigenschaft steht der Name des Rechners, auf dem die Anwendung, die das TAPI-Gerät verwendet, läuft.

Die Eigenschaft kann nur gelesen werden.

## Kontakt

---

**TapiOwnerRequest**

**TAPI-Gerät mit Besitzrechten**

**Typ** logic

**Siehe** Liste, Objekte

Mit dieser Eigenschaft kann bestimmt werden, ob die Applikation, die das TAPI-Gerät überwacht, versucht hat Verbindungen mit Besitzrechten (true) zu öffnen, oder nicht (false).

Hat eine Applikation Besitzrechte an den Verbindungen eines TAPI-Geräts, kann sie aktiv die Verbindungen verändern (zum Beispiel um Gespräche zu vermitteln oder Telefonkonferenzen zu initiieren). Ohne Besitzrechte ist lediglich eine Überwachung (Monitoring) der Verbindung möglich.

## Kontakt

TapiUserName



Betriebssystem Benutzerkontext

Typ [alpha\(256\)](#)

Siehe [Liste](#), [Objekte](#), [TapiInfo\(\)](#)

In dieser Eigenschaft steht der Name des Benutzers, unter dem ein Programm läuft, das ein TAPI-Gerät verwendet. Die Eigenschaft kann nur gelesen werden.

## Kontakt

---

### TerminalSession

Client läuft in einem Terminal

Typ logic  
Siehe Liste, Objekte,

Bei dem Objekt System kann ermittelt werden, ob der Client in einer Terminal-Sitzung gestartet wurde.

Die Eigenschaft liefert true zurück, wenn der CONZEPT 16-Client in einer Terminalsitzung gestartet wurde. In einem solchen Fall kann zum Beispiel die Ausgabe von Bildern unterdrückt werden, um lange Übertragungszeiten zu vermeiden.

### TerminalSessionID

#### ID der aktuellen Terminal-Sitzung

Typ int

Siehe Liste, Objekte, SysPropGet()

Bei dem Sys-Objekt kann über diese Eigenschaft die ID der aktuellen Terminal-Sitzung ermittelt werden. Wird diese Eigenschaft außerhalb einer Terminal-Sitzung abgefragt, ist deren Inhalt 0.

Die ID der Terminal-Sitzung kann verwendet werden, um unterschiedliche temporäre Verzeichnisse oder Dateien für die einzelnen Anwender eines Terminalservers anzulegen. Windows-Druckertreiber in einer Terminal-Sitzung werden vom System ebenfalls mit der Sitzungs-ID versehen.

## Kontakt

**TimeExternal**



**Zeitpunkt der letzten Änderung der externen Datei**

**Typ** caltime

**Siehe** Liste, Objekte, SysPropGet()

Diese Eigenschaft enthält das Datum und die Uhrzeit der letzten Änderung der externen Datei vor Import. Beim Export des binären Objektes wird der Änderungszeitpunkt der externen Datei wiederhergestellt. Die Eigenschaft kann nur gelesen werden.

### Type

Typ des Objekts oder Objektwerts

Typ int

Siehe Liste, Objekte, Datentypen

Die Eigenschaft hat abhängig vom Objekt unterschiedliche Bedeutungen und Inhalte.

- CteNode-Objekte
- Storage-Objekte
- Chart-Objekte
- CteNode

Bei diesem Objekt steht in der Eigenschaft der Datentyp des Objektwerts. Die Eigenschaft kann gelesen und geschrieben werden. Der Wert kann mit den \_Type...-Konstanten verglichen werden. Beim Setzen der Eigenschaft wird die Eigenschaft Value... auf NULL gesetzt, d. h. geleert.

Der Typ wird beim Setzen der Eigenschaft Value... automatisch auf den Datenyp des Wertes gesetzt.

Bei den Dynamischen Strukturen verfügt lediglich das CteNode-Objekt über die Eigenschaft Type.

- Storage-Objekte

In der Eigenschaft wird der Typ des Storage-Objekts angegeben. Die Eigenschaft kann nur gelesen werden. Der Wert besteht aus einer Kombination aus \_StoClass...- und \_StoType...-Konstanten. Folgende Konstanten stehen zur Verfügung:

- ◆ Fenster-Objekte (\_StoClassFrame)
  - \_StoTypeFrameFrame
  - \_StoTypeAppFrame      AppFrame
  - \_StoTypeMdiFrame      MdiFrame
  - \_StoTypeTrayFrame TrayFrame
- ◆ Menü-Objekte (\_StoClassMenu)
  - \_StoTypeMenu Menü
- ◆ Druck-Objekte (\_StoClassPrint)
  - \_StoTypePrintDoc PrintDoc
  - \_StoTypePrintDocRecord PrintDocRecord
  - \_StoTypePrintForm PrintForm
  - \_StoTypePrintFormList PrintFormList
- ◆ Bild-Objekte (\_StoClassPicture)
  - \_StoTypePicture      Rastergrafiken
  - \_StoTypeIcon      Symbole
  - \_StoTypeTile      Kachelgrafiken für Schaltflächen- und Listenobjekte
  - \_StoTypeTileMenu Kachelgrafiken für Menüobjekte
  - \_StoTypeTileTree      Kachelgrafiken für Baumobjekte
- ◆ Meta-Bilder (\_StoClassMeta)

## Kontakt

\_StoTypeEmf Vektorgrafiken

\_StoTypeAmf Vektorgrafiken

\_StoTypeWmf Vektorgrafiken

- ◆ Nicht klassifizierte Objekte

\_StoTypeObject Der genaue Typ ist nicht verfügbar.

Mit Hilfe der \_StoClass...-Konstanten kann die Gruppe ermittelt werden zu der das Objekt gehört. Der genaue Typ kann mit den \_StoType...-Konstanten bestimmt werden.

**Beispiele:**

```
if ((tStoObj->spType & _StoClassFrame) != 0) // Frame-Objektif (tStoObj->spType = _StoType
```

- Chart

Die Eigenschaft gibt den Typ der Grafik zurück. Der Inhalt kann mit den Konstanten ChartPie und ChartXY verglichen werden, um festzustellen, ob es sich um ein Torten- oder ein Koordinaten-Diagramm handelt. Bei diesem Objekt kann die Eigenschaft nur gelesen werden.

## Kontakt

TypeFile (Systemeigenschaft)

Dateityp des Storage-Objektes Typ

alpha

Liste, Objekte,

Siehe SysPropGet(),

SysPropSet()

Mit dieser Eigenschaft kann der Dateityp von Storage-Objekten ermittelt werden. Die Eigenschaft enthält je nach Typ folgende Werte:

Resource – 'rsc'

Tile – 'bmp'

Icon – 'ico'

Picture – 'bmp', 'jpg', 'tif', 'png', 'gif'

MetaPicture – 'emf', 'wmf'

### TypeMime

MIME-Typ des Objekts

Typ alpha(60)

Liste, Objekte,

Siehe SysPropGet(),

SysPropSet()

In dieser Eigenschaft kann der MIME-Typ des Objektinhalts abgelegt werden. Der Wert kann maximal 60 Stellen lang sein.

Über den MIME-Type kann genauer festgehalten werden, um welche Art von binärem Objekt es sich handelt. Der MIME-Typ besteht aus einem Mediumtyp und einem Subtyp, die durch einen Schrägstrich voneinander getrennt werden.

Beispiele:

```
'text/plain' 'text/html' 'text/xml' 'image/gif' 'image/jpeg' 'video/mpeg' 'application/msword'
```

Die Medientypen und Subtypen werden von der IANA (Internet Assigned Numbers Authority) standardisiert.

Die Eigenschaft wird von CONZEPT 16 in der BLOB-Verwaltung ausgewertet. Für Informationen, die über den MIME-Typen hinausgehen, kann zusätzlich die Eigenschaft TypeUser verwendet werden.

## Kontakt

---

### TypeUser

#### Benutzerdefinierte Typinformation

Typ alpha(60)  
Liste, Objekte, SysPropGet(),  
Siehe

In dieser Eigenschaft kann eine benutzerdefinierte Typinformation für den Objektinhalt abgelegt werden. Der Wert kann maximal 60 Stellen lang sein.

Diese Eigenschaft wird nicht von CONZEPT 16 ausgewertet und kann zur Speicherung von zusätzlichen Informationen zu dem binären Objekt verwendet werden. Hier kann zum Beispiel für ein Word-Dokument die genaue Word-Version abgelegt werden, nachdem in der Eigenschaft TypeMime bereits festgehalten ist, dass es sich um ein Word-Dokument handelt.

## Kontakt

**URI** Uniform-Resource-Identifier der HTTP-Anfrage

**Typ** alpha(4095)

Siehe Liste, Objekte, SysPropGet(), SysPropSet()

Eigenschaft des HTTP-Objekts. In dieser Eigenschaft wird der Uniform Resource Identifier (URI) der HTTP-Anfrage abgelegt. Die Eigenschaft steht bei HTTP-Objekten der Typen HttpRecvRequest und HttpSendRequest zur Verfügung. Sie kann bei dem Objekt-Typ HttpRecvRequest nur gelesen werden.

Wird ein neues Objekt vom Typ HttpSendRequest erzeugt

(HttpOpen( HttpSendRequest, ...)), wird die Eigenschaft mit '/' vorbelegt.

## Kontakt

Validated



Zeitpunkt der letzten Validierung

Typ caltime

Siehe Liste, Objekte, ValidatedUser

Diese Eigenschaft enthält Datum und Uhrzeit der letzten Validierung des Objekts. Der Zeitpunkt wird vom Validierungs-Editor neu gesetzt, wenn ein Objekt validiert wird.

## Kontakt

ValidatedUser



Benutzer, der das Objekt zuletzt validiert hat

Typ [alpha\(20\)](#)

Siehe [Liste](#), [Objekte](#), [Validated](#)

Diese Eigenschaft enthält den Namen des Benutzers, der das Objekt zuletzt validiert hat. Der Benutzername wird vom [Validierungs-Editor](#) neu gesetzt, wenn ein Objekt validiert wird.

## Kontakt

Value...

Wert des Objekts

Typ var

Siehe [Liste, Objekte,](#)

[CteNodeValueAlpha\(\)](#)

Wert des Objekts. Für jeden Datentyp existiert eine eigene Eigenschaft:

- ValueAlpha
- ValueBigInt
- ValueCaltime
- ValueColor
- ValueDate
- ValueDecimal
- ValueEvent
- ValueFloat
- ValueFont
- ValueInt
- ValueLogic
- ValuePoint
- ValueRange
- ValueRect
- ValueRtfTab
- ValueTime

Wird die Eigenschaft beschrieben, wird automatisch die Eigenschaft Type auf den entsprechenden Datentyp gesetzt. Soll der Wert gelesen werden, kann der Datentyp zuvor über diese Eigenschaft ermittelt werden.



Wird eine Value...-Eigenschaft von einem falschen Datentyp (!= Type) abgefragt, wird der Laufzeitfehler ErrFldType generiert.

Bei den Dynamischen Strukturen verfügt lediglich das CteNode-Objekt über die Eigenschaft Value.

Die Eigenschaft ValueAlpha kann Zeichenketten mit maximal 65.520 Zeichen enthalten. Beim Setzen kann sie aus mehreren Zeichenketten zusammengesetzt oder beispielsweise aus einem Memory-Objekt gelesen werden:



Längere Zeichenketten können mit CteNodeValueAlpha() gelesen und gesetzt werden.

```
// Inhalt eines Memory-Objektes in ValueAlpha schreibt XmlNode->spValueAlpha # tMem->MemReadStr(
```

## Kontakt

---

**Version (Systemeigenschaft)**

**TAPI-Version des TAPI-Gerätetreibers**

**Typ alpha(80)**

**Siehe Liste, Objekte, SysPropGet()**

**In dieser Eigenschaft steht die TAPI-Version des Gerätetreibers. Sie kann nur ausgelesen werden.**

**Von CONZEPT 16 können nur Gerätetreiber verwendet werden, die eine TAPI-Version von mindestens 1.4 besitzen.**

**Diese Version sollte nicht mit der Version der TAPI-Schnittstelle des Betriebssystems verwechselt werden. Die Version der Schnittstelle muss mindestens 2.0 entsprechen. Die Version 2.0 wird seit Microsoft Windows 98 standardmäßig mit dem Betriebssystem ausgeliefert.**

## Kontakt

Ermitteln und Setzen von Eigenschaften

Aufteilung der Eigenschaften in Gruppen

WinPropGet(),

WinPropSet(),

PrtPropGet(),

Siehe PrtPropSet(),

SysPropGet(),

SysPropSet(),

UrmPropGet(),

UrmPropSet()

Die Eigenschaften sind in vier Gruppen aufgeteilt:

- Eigenschaften von Oberflächen-Objekten
- Eigenschaften von Druck-Objekten
- Eigenschaften von System-Objekten
- Eigenschaften des Benutzersystems

Jede Gruppe besitzt ihre eigenen Befehle Win-, Prt-, Sys- und UrmPropGet/Set mit denen die Eigenschaft ausgelesen bzw. gesetzt werden kann. Das Ansprechen von Eigenschaften in Prozeduren funktioniert prinzipiell wie zur Entwurfszeit im Designer. Ist der Name der Eigenschaft bekannt, kann durch eine Objektreferenz und dem Namen der Eigenschaft, die Eigenschaft bei dem Objekt verändern oder auslesen.

Der Name der Eigenschaft wird für den Zugriff je nach Gruppe um \_WinProp, \_PrtProp oder \_SysProp erweitert. Die Eigenschaften von Benutzer-Objekten können nur über die \_UrmProp-Konstanten angesprochen werden.

Beispiele:

Setzen der Eigenschaft Caption bei einem Oberflächen-Objekt:

```
tErg # tHdlFrame->WinPropSet(_WinPropCaption, 'Suchen');
```

Setzen der Eigenschaft Caption bei einem Druck-Objekt:

```
tErg # tObjPrtText->PrtPropSet(_PrtPropCaption, 'Suchen');
```

Auslesen des temporären Pfades des Betriebssystems aus dem Sys-Objekt:

```
tErg # _Sys->SysPropGet(_SysPropTempPath, tTemp);
```

Auslesen des Namens eines Benutzers:

```
tErg # tHdlUser->UrmPropGet(_UrmPropLastname, tRealLastName);
```

In den Beispielen wird der logische Rückgabewert der Befehle nicht ausgewertet. Über diesen Rückgabewert kann festgestellt werden, ob das Lesen oder Setzen einer Eigenschaft erfolgreich war.

Zum Setzen und Lesen von Eigenschaften gibt es noch eine kürzere Schreibweise. In dieser Schreibweise wird vor dem Namen der Eigenschaft ein Kürzel wp, pp

## Kontakt

oder sp gesetzt, um zu kennzeichnen, ob es sich um eine Oberflächen-, Druck-, oder System-Eigenschaft handelt. Ein Kürzel für Eigenschaften des Benutzersystems existiert nicht. Die oben angeführten Beispiele können auch wie folgt geschrieben werden:

```
tHdlFrame->wpCaption # 'Suchen'; tObjPrtText->ppCaption # 'Suchen'; tTemp # _Sys->spTempPath;
```

Diese Anweisungen geben keinen Rückgabewert zurück, der den Erfolg der Operation darstellt. Kann eine Eigenschaft nicht gesetzt oder gelesen werden, erfolgt ein entsprechender Laufzeitfehler.

Die Befehle sollten immer dann verwendet werden, wenn das Vorhandensein einer Eigenschaft zu einem Objekt nicht sichergestellt ist. Zum Beispiel in einer Funktion, die ein Objekt übergeben bekommt. In den anderen Fällen kann die kurze Schreibweise verwendet werden.

## Kontakt

### Ereignisse

Über Ereignisse wird eine Applikation gesteuert

#### Alphabetische

Siehe Liste,

#### Ablaufpläne

Ereignisse treten immer dann auf, wenn der Anwender eine Aktion ausführt. Eine Aktion ist das Drücken einer Schaltfläche, Maustaste, das Aufrufen eines Menüpunktes usw.

Bei jedem Ereignis kann eine Prozedur bzw. Prozedurfunktion aufgerufen werden.

Die Prozeduren werden bei dem betreffenden Objekt im Eigenschaftsfenster auf der Seite Ereignisse eingetragen. Zur Laufzeit können Ereignisfunktionen mit den Anweisungen WinEvtProcNameSet() und WinEvtProcNameGet() eingetragen und ermittelt werden. Die Ausführung von Ereignissen kann zur Laufzeit mit der Anweisung WinEvtProcessSet() ausgesetzt werden.

Die Ereignisse untergliedern sich in folgende Bereiche:

- Init (Initialisierungs-Ereignisse)
- Term (Terminierungs-Ereignisse)
- Focus (Fokus-Ereignisse)
- Mouse (Maus-Ereignisse)
- Menu (Menü-Ereignisse)
- Key (Tastatur-Ereignisse)
- Placement (Positionierungs-Ereignisse)
- System (System-Ereignisse)
- Special (Objektspezifische Ereignisse)
- Drag & Drop-Ereignisse

**Init (Initialisierungs-Ereignisse)**

Liste sortiert

nach

Gruppen,

Alphabetische

Liste aller

Siehe

- EvtCreated

- EvtInit

## Kontakt

### EvtInit

#### Laden des Objekts

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
		<b>Wird nicht ausgewertet</b>

#### Resultat logic

ausgewertet  
Liste, Objekte, EvtTerm,

Siehe

Ereignisbefehle, Beispiel

Die angegebene Prozedur wird ausgeführt, wenn das Objekt geladen wurde. In dieser Prozedur kann beispielsweise ein globaler Datenbereich für das Fenster angelegt werden.

#### Definition des Funktionskopfes:

```
sub EvtInit( aEvt : event; // Ereignis) : logic; //
```

#### aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

#### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.



Das Ereignis kann bei dynamisch erzeugten Objekten nicht mit WinEvtProcNameSet() gesetzt werden.

#### Beispiel:

```
sub EvtInit( aEvt : event; // Ereignis) : logic; // Wird nicht ausgewertet{ // Datenbereic
```

## Kontakt

### EvtCreated

Ereignis nach der Anzeige eines Objekts

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>

Die angegebene Prozedur wird ausgeführt, nachdem das Fenster-Objekt und alle untergeordneten Objekte erstellt und auf dem Bildschirm dargestellt wurden. Hier können Initialisierungen nach der Anzeige des Objekts vorgenommen werden.

Definition des Funktionskopfes:

```
sub EvtCreated( aEvt : event;      // Ereignis): logic;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel:

```
sub EvtCreated( aEvt : event;      // Ereignis): logic;
```

**Term (Terminierungs-Ereignisse)**

Liste sortiert

nach

Gruppen,

Alphabetische

Liste aller

Siehe

- EvtClose

- EvtTerm

## Kontakt

### EvtClose

#### Schließen eines Fensters

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>Resultat</u>	<u>logic</u>	Fenster schließen?
		<u>Liste, Objekte, EvtTerm,</u>

Siehe Ereignisbefehle

Die angegebene Prozedur wird ausgeführt, wenn der Benutzer das Fenster schließen möchte. Über den Rückgabewert der Funktion wird entschieden, ob das Fenster geschlossen wird oder nicht. Bei true als Rückgabewert wird das Fenster geschlossen, bei false wird es nicht geschlossen.

#### Definition des Funktionskopfes:

```
sub EvtClose( aEvt : event; // Ereignis) : logic;
```

#### aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

#### Resultat

Liefert die Prozedur false als Rückgabewert wird das Fenster nicht geschlossen.

#### Beispiel

```
sub EvtClose( aEvt : event; // Ereignis) : logic; // Fenster schließen?
```

## Kontakt

## EvtTerm

## Entladen des Objekts

Name	Typ	Beschreibung
<u>aEvt</u>	event	Ereignisinformation
Resultat	logic	Wird nicht

---

ausgewertet

### Liste, Objekte, EvtClose,

**Siehe** [EvtInit](#), [Ereignisbefehle](#),  
[Beispiel](#)

**Die angegebene Prozedur wird ausgeführt, wenn das Objekt entladen wurde.**

An dieser Stelle kann die Freigabe **globaler Datenbereiche**, die für dieses Objekt gültig sind, vorgenommen werden. Bei der Verwendung von **MDI-Frames** und der Eigenschaft **DbVar** müssen die globalen Datenbereiche nicht gelöscht werden. Zu dem Zeitpunkt, zu dem das Ereignis **EvtTerm** ausgeführt wird, ist der Variablenbereich bereits entfernt und (wenn vorhanden) bereits der Variablenbereich eines anderen MDI-Fensters instanziert. Soll vor dem Schließen eines MDI-Fensters noch auf den Datenbereich zugegriffen werden, muss das im Ereignis **EvtClose** erfolgen.

## **Definition des Funktionskopfs:**

```
sub EvtTerm(    aEvt           : event;      // Ereignis): logic;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

## Resultat

**Der Rückgabewert der Funktion wird nicht ausgewertet.**



**Das Ereignis kann bei dynamisch erzeugten Objekten nicht mit WinEvtProcNameSet() gesetzt werden.**

## Beispiel:

```
sub EvtTerm( aEvt : event; // Ereignis ) : logic // Wird nicht ausgewertet
```

**Focus (Fokus-Ereignisse)**

Liste sortiert

nach

Gruppen,

Alphabetische

Liste aller

Siehe

- EvtChangedActive
- EvtFocusCancel
- EvtFocusInit
- EvtFocusTerm
- EvtMdiActivate

## Kontakt

### EvtChangedActive

Ereignis wenn sich das aktive GroupTile-Objekt ändert.

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aActiveOld</u>	<u>handle</u>	Deskriptor des deaktivierten <small>GroupTile-Objekts</small>
<u>aActiveNew</u>	<u>handle</u>	Deskriptor des aktivierten <small>GroupTile-Objekts</small>
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
<u>Siehe</u>	<u>Liste, Objekte, Ereignisbefehle</u>	

Die Funktion wird aufgerufen, wenn sich das aktive GroupTile-Objekt ändert.

Definition des Funktionskopfes:

```
sub EvtChangedActive( aEvt : event; // Ereignis aActiveOld : handle
```

**aEvt**

In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aActiveOld**

Hier wird der Deskriptor des GroupTile-Objekts übergeben, dass den Fokus verloren hat.

**aActiveNew**

Hier wird der Deskriptor des GroupTile-Objekts übergeben, dass den Fokus bekommen hat.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

## Kontakt

### EvtFocusInit

Ereignis nach dem Erlangen des Eingabe-Fokus.

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aFocusObject</u>	<u>handle</u>	<b>Objekt, das den Fokus</b>
<u>Resultat</u>	<u>logic</u>	zuvor hatte Wird nicht ausgewertet
Siehe		<u>Liste, Objekte, EvtFocusTerm,</u> <u>Ereignisbefehle</u>

Die Funktion wird aufgerufen, nach dem das Objekt den Eingabe-Fokus erlangt hat.

Definition des Funktionskopfes:

```
sub EvtFocusInit(  aEvt : event;      // Ereignis  aFocusObject : handle;
```

**aEvt**

Die Funktion hat einen Übergabeparameter vom Typ event und vom Typ handle. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aFocusObject**

aFocusObject enthält den Deskriptor des Objekts, das zuvor den Fokus hatte. Der Übergabeparameter hat den Wert 0, wenn das Quellobjekt nicht ermittelt werden konnte, weil es zum Beispiel in einer anderen Applikation liegt.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.



Innerhalb dieses Ereignisses dürfen keine Dialoge oder Messageboxen geöffnet werden. Der Fokus würde in den Dialog gesetzt werden. Nach dem Verlassen des Dialogs, wird der Fokus wieder in das ursprüngliche Objekt gesetzt und das Ereignis würde wiederum aufgerufen werden.

## Kontakt

### EvtFocusTerm

Ereignis vor dem Verlassen des Objekts

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aFocusObject handle		Objekt, das den Fokus erhalten wird
Resultat	logic	Eingabeobjekt verlassen? Liste, Objekte, EvtFocusInit,
Siehe		EvtFocusCancel, Ereignisbefehle

Die Funktion wird unmittelbar, nachdem das Objekt den Eingabe-Fokus verloren hat, aufgerufen.

Definition des Funktionskopfes:

```
sub EvtFocusTerm( aEvt : event; // Ereignis aFocusObject : handle;
```

aEvt

Die Funktion hat einen Übertragungsparameter vom Typ **event** und einen Übertragungsparameter vom Typ **handle**. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aFocusObject

aFocusObject enthält den Deskriptor des Objekts, das als nächsten den Fokus erhalten wird. Der Übertragungsparameter hat den Wert 0, wenn das Zielobjekt nicht ermittelt werden konnte, weil es zum Beispiel in einer anderen Applikation liegt.

Resultat

Mit Hilfe des Rückgabewerts bei diesem Ereignis kann in einem Eingabe-Objekt verblieben werden. Bei false als Rückgabewert darf das Objekt nicht verlassen werden und das Ereignis EvtFocusCancel wird ausgelöst. Bei true darf das Objekt verlassen werden.

 Falls abgefragt werden möchte, welche Taste zum Verlassen des Objekts verwendet wurde, kann dies mit Hilfe von WinInfo(..., WinFocusKey) in diesem Ereignis ermittelt werden.

 Über die Ereignisse EvtFocusTerm und EvtFocusCancel kann eine individuelle Feldkontrolle eingerichtet werden. Üblicherweise wird aber eine Kontrolle der Eingaben beim Verlassen eines Dialogs oder bei einer anderen Benutzeraktion durchgeführt. Es ist zu beachten, dass das Ereignis EvtFocusTerm auch beim Taskwechsel aufgerufen wird. Innerhalb dieses Ereignisses dürfen keine Dialoge oder Messageboxen aufgerufen werden. Der Fokus wird in den Dialog übertragen. Nach dem Schließen des Dialogs steht der Fokus wieder in seinem ursprünglichen Objekt und das Ereignis würde wieder aufgerufen werden, sobald das Objekt wieder verlassen wird.

### EvtFocusCancel

Verbleib im Eingabe-Objekt

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen Wird nicht

#### Resultat logic

ausgewertet

Liste, Objekte,

Siehe EvtFocusTerm,

Ereignisbefehle

Dieses Ereignis wird nur dann ausgelöst, wenn das Objekt nicht verlassen werden darf, also das Ereignis EvtFocusTerm den Wert false zurückgibt. An dieser Stelle kann eine Messagebox (zum Beispiel mit WinDialogBox()) oder ein Dialog angezeigt werden, der erklärt, warum das Objekt nicht verlassen werden darf.



Über die Ereignisse EvtFocusTerm und EvtFocusCancel kann eine individuelle Feldkontrolle eingerichtet werden. Üblicherweise wird aber eine Kontrolle der Eingaben beim Verlassen eines Dialogs oder bei einer anderen Benutzeraktion durchgeführt. Es ist zu beachten, dass das Ereignis EvtFocusTerm auch beim Taskwechsel aufgerufen wird.

Definition des Funktionskopfes:

```
sub EvtFocusCancel( aEvt : event; // Ereignis) : logic;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

## Kontakt

### EvtMdiActivate

#### Aktivierung eines MDI-Frames

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen

**Wird nicht**

#### Resultat logic

ausgewertet

Liste, Objekte,

Siehe Ereignisbefehle,

Ereignisabläufe MdiFrame

Das Ereignis wird ausgelöst, wenn zu diesem MDI-Fenster gewechselt wird. Mit Hilfe dieses Ereignis können beispielsweise die Menüeinträge des Applikationsfensters dem aktuellen MdiFrame angepasst werden.

Definition des Funktionskopfes:

```
sub EvtMdiActivate( aEvt : event; // Ereignis): logic;
```

**aEvt**

Im Parameter aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts übergeben. Dies ist gleichzeitig der MdiFrame, der aktiviert werden soll.

**Resultat**

Der Rückgabewert wird bei diesem Ereignis nicht ausgewertet.

**Mouse (Maus-Ereignisse)**

Liste sortiert

nach

Gruppen,

Siehe Alphabetische

Liste aller

Ereignisse

- EvtIvlDropItem

- EvtMouse

- EvtMouseItem

- EvtMouseMove

**EvtIvlDropItem****Drop-Vorgang**

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aHdlTarget</u>	<u>handle</u>	Deskriptor des Ziel-Objekts ( <u>GanttGraph</u> )
<u>aHdlIvl</u>	<u>handle</u>	Deskriptor des Intervalls
<u>aDropType</u>	<u>int</u>	Drop-Typ
<u>aRect</u>	<u>rect</u>	Position des Intervalls
<u>Resultat</u>	<u>logic</u>	Drop-Vorgang durchführen? Liste, Objekte,

Siehe

Ereignisbefehle

Das Ereignis wird bei einem GanttGraph-Objekt ausgelöst, wenn ein Intervall oder ein Box-Objekt vom Anwender verschoben, kopiert oder in der Größe verändert werden soll.

**Definition des Funktionskopfes:**

```
sub EvtIvlDropItem( aEvt : event; // Ereignis aHdlTarget : handle;
```

**aEvt**

In dem Übergabeparameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

In der Komponente ID des Übergabeparameters aEvt wird

WinEvtIvlDropItemOverlap übergeben, wenn das Intervall so vergrößert, verschoben oder kopiert wurde, dass es sich mit einem anderen Intervall überschneidet.

**aHdlTarget**

Bei aHdlTarget handelt es sich um den Deskriptor des GanttGraph-Objekts, in dem das Intervall fallengelassen wird. aHdlTarget kann ungleich aEvt:Obj sein, da bei einer Verschiebe- oder Kopieren-Operation des Intervalls, die zugehörigen GanttGraphen unterschiedliche Objekte sein können. aEvt:Obj ist der Deskriptor des Quell-Gantt-Objekts, in dem der Vorgang gestartet wurde.

**aHdlIvl**

aHdlIvl ist des Deskriptor des Intervalls bzw. Box-Objektes, das verschoben, kopiert oder in der Größe verändert werden soll.

**aDropType**

aDropType kann folgende Werte annehmen:

## Kontakt

WinIvlDropSizeLeft

Linke Kante des Intervalls bzw. Box-Objektes wurde gezogen

WinIvlDropSizeRight

Rechte Kante des Intervalls bzw. Box-Objektes wurde gezogen

WinIvlDropSizeTop

Obere Kante des Box-Objektes wurde gezogen

WinIvlDropSizeBottom

Untere Kante des Box-Objektes wurde gezogen

WinIvlDropSizeLeftTop

Linke obere Ecke des Box-Objektes wurde gezogen

WinIvlDropSizeLeftBottom

Linke untere Ecke des Box-Objektes wurde gezogen

WinIvlDropSizeRightTop

Rechte obere Ecke des Box-Objektes wurde gezogen

WinIvlDropSizeRightBottom Rechte untere Ecke des Box-Objektes wurde gezogen

WinIvlDropMove

Intervall wurde verschoben

WinIvlDropCopy

Intervall wurde kopiert

aRect

Über aRect (vom Typ rect) kann die Position des Intervalls nach dem Verschieben, Kopieren oder Größenänderung ermittelt werden.

Resultat

Über den Rückgabewert kann entschieden werden, ob der Drop-Vorgang durchgeführt werden soll (true) oder nicht (false).

Das Intervall bzw. die Box kann nur im Bereich des Rasters verändert werden. Sobald das Objekt das Raster verlässt, wird der Mauszeiger auf den NoDrop-Zeiger verändert. Ist die Eigenschaft OleDropMode auf WinOleDynamic gesetzt, findet keine Änderung des Mauszeigers statt. Soll der Mauszeiger trotzdem verändert werden, muss in der Eigenschaft GanttFlags des GanttGraph-Objektes die Ausprägung WinGanttDropLimitBound gesetzt werden.

Beispiel:

```
sub EvtIvlDropItem( aEvt : event; // Ereignis aHdlTarget : handle; // Ziel-o
```

## Kontakt

EvtMouse

Maustaste

Name      Typ      Beschreibung  
aEvt    event Ereignisinformationen

aButton int      Maustaste

                Ereignis an  
Resultat logic      auslösendes Objekt

                weitergeben?  
Siehe                    Liste, Objekte,

Ereignisbefehle

Die angegebene Prozedur wird ausgeführt, wenn mit einer Maustaste auf das Objekt geklickt wurde.



Beim Objekt Statusbar wird das Ereignis nicht ausgeführt, wenn in dem Ereignis EvtMouseItem eine Funktion eingetragen ist.

Definition des Funktionskopfes:

```
sub EvtMouse( aEvt : event; // Ereignis aButton : int; //
```

aEvt

In aEvt vom Typ event wird unter anderem der Deskriptor des Objekts übergeben, welches das Ereignis ausgelöst hat. Dies ist das Objekt, in dem die Ereignisfunktion eingetragen wurde.



Bei Scrollbox-Objekten bezieht sich aEvt:Pos auf die linke obere Ecke des Scrollbereiches und nicht des Ausgabebereiches.

aButton

aButton enthält die Taste oder die Tastenkombination, die das Ereignis auslöste:

WinMouseLeft      linke Maustaste

WinMouseRight      rechte Maustaste

WinMouseMiddle mittlere Maustaste

WinMouseMask      linke, rechte oder mittlere Maustaste

WinMouseDouble Doppelklick

WinMouseShift      -Taste

WinMouseCtrl      -Taste

Die Konstanten können miteinander kombiniert werden. Wurde zum Beispiel die

-Taste gedrückt und mit der linken Maustaste ein Doppelklick ausgeführt, steht im Übergabeparameter aButton der Wert WinMouseLeft | WinMouseDouble | WinMouseShift.

Über die Konstante WinMouseMask können zusätzlich gedrückte Tasten ausgeblendet werden (vgl. Beispiel).

Resultat

## Kontakt

Wird true zurückgegeben, wird das Ereignis an das auslösende Objekt weitergeleitet, ansonsten nicht.



Wird beim CtxDocEdit true zurückgegeben, wird anschließend das Ereignis EvtMouseItem ausgelöst.

Beispiele:

```
// Doppelklickif (aButton & _WinMouseDouble > 0) ...switch (aButton & _WinMouseMask) { // linke
```



Bei der Verwendung des Debuggers in Verbindung mit Breakpoints sowie den Anweisungen DbgStop() und DbgControl( DbgStop) kann kein Doppelklick ausgelöst werden. Bei einem Doppelklick werden immer zwei Ereignisse ausgelöst: Da das Windows System nach einem erfolgten Klick nicht weiß, ob innerhalb kurzer Zeit ein weiterer Klick erfolgt, wird zunächst ein Ereignis für den Klick und anschließend das Ereignis für den Doppelklick ausgelöst. Wird die Verarbeitung in dem Ereignis angehalten, vergeht zwischen den beiden Klicks zu viel Zeit, um sie noch als einen Doppelklick zu erkennen.

## Kontakt

### EvtMouseItem

#### Maus-Ereignis

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aButton</u>	<u>int</u>	Maustaste
<u>aHitTest</u>	<u>int</u>	Hittest-Code
<u>aItem</u>	<u>handle</u>	Deskriptor der Spalte oder des
<u>aID</u>	<u>int / bigint</u>	Intervall-Objekts RecID bei RecList / selektierte Zeile bei DataList / Zeile bei
<u>Resultat logic</u>		GanttGraph Ereignis an die Objekte RecList, GanttGraph oder RecNavigator
Siehe		weiterreichen <u>Liste, Objekte</u> , <u>Ereignisbefehle</u> , <u>Beispiel</u>

Dieses Ereignis wird aufgerufen, wenn innerhalb eines Objektes, das aus mehreren Bestandteilen bzw. untergeordneten Objekten besteht, geklickt wurde. Die Möglichkeiten gehen über die des Ereignisses EvtMouse hinaus.



Ist bei dem Objekt Statusbar das Ereignis gesetzt, wird das Ereignis EvtMouse nicht ausgeführt.

Das Ereignis wird sofort beim Drücken der Maustaste ausgelöst. Erfolgt also ein Doppelklick auf ein Objekt, wird zunächst das Ereignis für einen einfachen Klick und kurz darauf das Ereignis für den doppelten Klick aufgerufen.

Definition des Funktionskopfes:

```
sub EvtMouseItem( aEvt : event; // Ereignis aButton : int;
```

Alternativer Funktionskopf:

```
sub EvtMouseItem( aEvt : event; // Ereignis aButton : int;
```

aEvt

In aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts übergeben.

aButton

aButton gibt die Taste oder die Tastenkombination an, die das Ereignis auslöste:

<u>WinMouseLeft</u>	linke Maustaste
<u>WinMouseRight</u>	rechte Maustaste

## Kontakt

WinMouseMiddle mittlere Maustaste

WinMouseMask linke, rechte oder mittlere Maustaste

WinMouseDouble Doppelklick

WinMouseShift  Taste

WinMouseCtrl  Taste

Die Konstanten können miteinander kombiniert werden. Wurde zum Beispiel die

-Taste gedrückt und mit der linken Maustaste ein Doppelklick ausgeführt, steht im Übergabeparameter aButton der Wert WinMouseLeft | WinMouseDouble | WinMouseShift.

aHitTest

In diesem Parameter wird der Objekttyp angegeben, der an der Stelle der Mausposition vorliegt:

<u>WinHitNone</u>	kein Objekt
<u>WinHitLstHeader</u>	Spaltenkopf eines RecList-, DataList- oder RecView-Objektes
<u>WinHitLstView</u>	Spalte eines RecList- oder DataList-, Anzeigebereich eines DocView- oder leerer Bereich eines RecView-Objektes
<u>WinHitStatusbar</u>	Statusbar
<u>WinHitStatusBarButton</u>	Statusbar-Button-Objekt in einer Statusbar
<u>WinHitCorner</u>	Zwischenraum der Achsen (linke obere Ecke eines Gantt-Graphen)
<u>WinHitGanttView</u>	Ausgabebereich eines Gantt-Graphen
<u>WinHitIvl</u>	Interval-Objekt eines Gantt-Graphen
<u>WinHitIvlBox</u>	Box-Objekt eines Gantt-Graphen (_WinGanttIvlBoxClick muss in GanttFlags gesetzt sein)
<u>WinHitIvlLeft</u>	linke Seite des Intervalls eines Gantt-Graphen
<u>WinHitIvlRight</u>	rechte Seite des Intervalls eines Gantt-Graphen
<u>WinHitAxis</u>	Achse eines Gantt-Graphen
<u>WinHitTreeNode</u>	Knoten-Objekt eines TreeView
<u>WinHitRecNavFirst</u>	"ersten Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavLast</u>	"letzten Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavPrev</u>	"vorigen Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavNext</u>	"nächsten Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavPgUp</u>	"fünf Datensätze zurück blättern" eines RecNavigator-Objektes
<u>WinHitRecNavPgDn</u>	"fünf Datensätze vor blättern" eines RecNavigator-Objektes
<u>WinHitRecNavNew</u>	"Datensatz einfügen" eines RecNavigator-Objektes
<u>WinHitRecNavDel</u>	"Datensatz löschen" eines RecNavigator-Objektes
<u>WinHitRecNavSave</u>	"Datensatz Speichern" eines RecNavigator-Objektes
<u>WinHitRecNavLock</u>	"Datensatz sperren" eines RecNavigator-Objektes
<u>WinHitDocViewPage</u>	Seite eines DocView-Objektes
<u>WinHitPrtPage</u>	Seite eines PrtJobPreview-Objektes

## Kontakt

<u>WinHitGroupColumn</u>
<u>WinHitGroup</u>
<u>WinHitCanvasView</u>
<u>WinHitCanvasGraphic</u>

Spalte eines RecView-Objektes
Gruppe eines RecView-Objektes
Ausgabebereich eines Canvas-Objektes
CanvasGraphic eines Canvas-Objektes

Beim RecNavigator wird das Ereignis bei den Leseoperationen nach dem Lesen des Datensatzes und bei den Schreiboperationen vor der Datensatzoperation aufgerufen. Bei den Schreiboperationen kann durch die Rückgabe von false die Datensatzoperation verhindert werden.



Auf die Eigenschaften einer Gruppe eines RecView-Objektes kann NUR lesend zugegriffen werden.

aItem

Der Übergabeparameter enthält in Abhängigkeit von dem in aHitTest übergebenenden Wert den Deskriptor des angeklickten Objekts:

<u>aHitTest</u>
<u>WinHitNone</u>
<u>WinHitListView</u>
<u>WinHitListHeader</u>
<u>WinHitStatusbar</u>
<u>WinHitStatusBarButton</u> Deskriptor des Statusbar-Button-Objekts
<u>WinHitIvl</u>
<u>WinHitIvlLeft</u>
<u>WinHitIvlRight</u>
<u>WinHitIvlBox</u>
<u>WinHitGanttView</u>
<u>WinHitAxis</u>
<u>WinHitTreeNode</u>
<u>WinHitDocViewPage</u>
<u>WinHitPrtPage</u>
<u>WinHitGroupColumn</u>
<u>WinHitGroup</u>
<u>WinHitCanvasView</u>
<u>WinHitCanvasGraphic</u>

<u>aItem</u>
0
Deskriptor der Spalte, die angeklickt wurde
Deskriptor der Spalte, deren Spaltenkopf angeklickt wurde
0
Deskriptor des angeklickten Interval-Objektes
Deskriptor des Interval-Objektes, dessen linke Begrenzung angeklickt wurde
Deskriptor des Interval-Objektes, dessen rechte Begrenzung angeklickt wurde
Deskriptor des Box-Objektes, das angeklickt wurde
Die Nummer der angeklickten Spalte, die Zeile wird in aID übergeben.
Deskriptor der angeklickten Achse
Deskriptor des angeklickten Knoten-Objektes
0
Deskriptor des angeklickten Objektes auf der Seite
Deskriptor der Spalte, die angeklickt wurde
Deskriptor der Gruppe, die angeklickt wurde
0
Deskriptor des angeklickten CanvasGraphic-Objektes im Canvas-Objekt

Wurde das Ereignis bei einem PrtJobPreview-Objekt ausgelöst, kann der Name des Objekts (aItem->wpName) nur dann ermittelt werden, wenn der Druckjob zuvor mit der Option PrtJobOpenVerbose gespeichert wurde. Das PrtJobPreview-Objekt der Druckvorschau kann mit dem Befehl PrtSearch(..., 'PpvControl') ermittelt werden.

Wurde das Ereignis bei einem RecView-Objekt mit Hittest WinHitGroup ausgelöst, sind die Eigenschaften SelectorItem und SelectorSubItem entsprechend gesetzt, so

## Kontakt

dass auf die Eigenschaften des entsprechenden Items zugegriffen werden kann. Die Eigenschaften können nur gelesen werden.

### aID

Enthält aItem einen gültigen Deskriptor auf eine Spalte oder Gruppe, wird in aID entweder die Datensatz-ID des angeklickten Datensatzes (bei den Objekten RecList, RecListPopup und RecView) oder die Zeilennummer (bei den Objekten DataList und DataListPopup) übergeben.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

Wurde innerhalb eines Statusbar-Objekts geklickt, ist aID immer 0.

Enthält aItem einen gültigen Deskriptor auf ein Intervall, enthält aID den Wert, der in der Eigenschaft ID des Intervalls enthalten ist.

Enthält aItem einen gültigen Deskriptor auf eine Achse, wird in aID die Nummer der Skala-Zelle ( $>= 0$ ) übergeben. Liegt der Punkt, der zum Auslösen des Ereignisses führte, im Titel der Achse, wird in aID der Wert -1 übergeben.

Wird in aItem die Nummer einer Spalte eines Gantt-Graphen übergeben, befindet sich in diesem Parameter die angeklickte Zeile.

Wurde das Ereignis bei einem DocView-Objekt ausgelöst, steht in aID die Nummer der angeklickten Seite. Existiert an der geklickten Stelle keine Seite, wird 0 übergeben. Das Argument aItem ist immer 0.

Ist das Ereignis bei einem PrtJobPreview eingetragen, wird in aID die angeklickte Seitennummer übergeben.

### Resultat

Wird true zurückgegeben, wird das Ereignis an das Objekt (Reclist bzw. GanttGraph) weitergeleitet, ansonsten nicht. Beim RecNavigator-Objekt kann durch die Rückgabe von false eine Schreiboperation des Datensatzes verhindert werden.

### Besonderheiten



Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

### Beispiele

```
sub EvtMouseItem( aEvt : event; // Ereignis aButton : int; // Maustaste aHitTest :
```



Bei der Verwendung des Debuggers in Verbindung mit Breakpoints sowie den Anweisungen DbgStop() und DbgControl( DbgStop) kann kein Doppelklick

## **Kontakt**

**ausgelöst werden. Bei einem Doppelklick werden immer zwei Ereignisse ausgelöst: Da das Windows System nach einem erfolgten Klick nicht weiß, ob innerhalb kurzer Zeit ein weiterer Klick erfolgt, wird zunächst ein Ereignis für den Klick und anschließend das Ereignis für den Doppelklick ausgelöst. Wird die Verarbeitung in dem Ereignis angehalten, vergeht zwischen den beiden Klicks zu viel Zeit, um sie noch als einen Doppelklick zu erkennen. Soll die Verarbeitung von einem Doppelklick angehalten werden, darf der Breakpoint erst nach der Auswertung der Maustasten erfolgen.**

## Kontakt

### EvtMouseMove

#### Mausbewegung in ein Objekt

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
		Mausaktion
		<u>WinMouseMoveEnter</u>
		Maus wird in das Objekt bewegt
		<u>WinMouseMoveLeave</u>
		Maus wird aus dem Objekt bewegt
<u>aMove</u>	<u>int</u>	
		<u>WinMouseMoveChange</u> Maus wird innerhalb des Objekts bewegt
<u>aMouseBtn</u>	<u>int</u>	Maustaste (optional)
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte, Ereignisbefehle</u>

Die angegebene Prozedur wird ausgeführt, wenn der Mauszeiger in den Clientbereich des Objekts bewegt wird oder diesen wieder verlässt.

Die Verarbeitung des Ereignisses sollte so kurz wie möglich sein, da während der Durchführung keine Benutzerinteraktion möglich ist.

Definition des Funktionskopfes:

```
sub EvtMouseMove( aEvt : event; // Ereignis aMove : int;
```

aEvt

In aEvt vom Typ event wird unter anderem der Deskriptor des Objekts übergeben, welches das Ereignis ausgelöst hat. Dies ist das Objekt, in dem die Ereignisfunktion eingetragen wurde.

aMove

aMove enthält die Aktion, die durchgeführt wurde. Der Wert kann mit folgenden Konstanten verglichen werden:

- WinMouseMoveEnter Der Mauszeiger wurde in das Objekt bewegt.
- WinMouseMoveLeave Der Mauszeiger wurde aus dem Objekt bewegt.
- WinMouseMoveChange Der Mauszeiger wurde innerhalb des Objekts bewegt. aMouseBtn

## Kontakt

aMouseBtn gibt die Taste oder die Tastenkombination an, die aktiv waren, während das Ereignis ausgelöst wurde. Der Wert kann mit folgenden Konstanten verglichen werden:

<u>WinMouseLeft</u>	linke Maustaste
<u>WinMouseRight</u>	rechte Maustaste
<u>WinMouseMiddle</u>	mittlere Maustaste
<u>WinMouseMask</u>	linke, rechte oder mittlere Maustaste
<u>WinMouseShift</u>	 -Taste
<u>WinMouseCtrl</u>	 -Taste

Die Konstanten können miteinander kombiniert werden. Wurde zum Beispiel die

 -Taste gedrückt und mit der linken Maustaste geklickt, während die Maus bewegt wurde, steht im Übergabeparameter aMouseBtn der Wert WinMouseLeft | WinMouseShift.

## Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Beispiel:

```
sub EvtMouseMove( aEvt : event;      // Ereignis aMove : int;
```

**Menu (Menü-Ereignisse)**

Liste sortiert

nach

Gruppen,

Siehe

Alphabetische

Liste aller

- EvtMenuCommand

- EvtMenuContext

- EvtMenuInitPopup

- EvtMenuPopup

## Kontakt

### EvtMenuCommand

Menüoption oder Toolbar-Button ausgewählt

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aMenuItem handle	auslösender	Menüpunkt / Toolbar-Button

Resultat logic Wird nicht ausgewertet

Siehe Liste, Objekte, Ereignisbefehle

Dieses Ereignis wird ausgeführt, wenn eine Schaltfläche einer Toolbar, ein Menüpunkt im Fenster oder ein Menüpunkt innerhalb eines Kontextmenüs angeklickt wurde.

 Damit das Ereignis beim Klick auf einen Toolbar-Button ausgelöst wird, muss das Ereignis beim Frame-Objekt angegeben werden. Wird das Ereignis bei der Toolbar angegeben, reagiert es nur für das Kontextmenü.

Definition des Funktionskopfes:

```
sub EvtMenuCommand( aEvt : event; // Ereignis aMenuItem : handle;
```

aEvt

Im Parameter aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes übergeben. Das Objekt entspricht demjenigen, in dem die Ereignisfunktion eingetragen wurde.

aMenuItem

aMenuItem enthält den Deskriptor des ausgewählten Menüpunktes oder der ausgewählten Schaltfläche.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel:

```
sub EvtMenuCommand( aEvt : event; // Ereignis aMenuItem : handle; // Menüeintrag /
```

## Kontakt

### EvtMenuContext

Kontextmenü aktiviert

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
<u>aHitTest</u>	int	Hittest-Code
<u>aItem</u>	handle	Spalte oder
		Gantt-Intervall
<u>aID</u>	int / bigint	Datensatz-ID bei <u>RecList</u> , Zeile bei <u>DataList</u>
<u>Resultat logic</u>		Wird nicht ausgewertet

Siehe Liste, Objekte, Ereignisbefehle

Die angegebene Prozedur wird ausgeführt, bevor das Kontextmenü geladen und angezeigt wird. Innerhalb der Ereignisprozedur kann nach Bedarf ein anderes Menü über die Eigenschaft MenuNameCntxt gesetzt werden.

Definition des Funktionskopfes:

```
sub EvtMenuContext( aEvt : event; // Ereignis aHitTest : int;
```

Alternativer Funktionskopf:

```
sub EvtMenuContext( aEvt : event; // Ereignis aHitTest : int;
```

aEvt

In aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes übergeben.

### aHitTest

In diesem Parameter wird der Objekttyp angegeben, der an der Stelle der Mausposition vorliegt:

<u>WinHitNone</u>	kein Objekt
<u>WinHitLstHeader</u>	Spaltenkopf eines <u>RecList</u> -, <u>DataList</u> - oder <u>RecView</u> -Objektes
<u>WinHitLstView</u>	Spalte eines <u>RecList</u> - oder <u>DataList</u> -, Anzeigebereich eines <u>DocView</u> - oder leerer Bereich eines <u>RecView</u> -Objektes
<u>WinHitCorner</u>	Zwischenraum der Achsen ( <u>linke obere Ecke</u> )
<u>WinHitGanttView</u>	Ausgabebereich eines <u>Gantt</u> -Graphen
<u>WinHitIvl</u>	<u>Intervall</u> -Objekt eines <u>Gantt</u> -Graphen
<u>WinHitIvlLeft</u>	linke Seite des <u>Intervalls</u> eines <u>Gantt</u> -Graphen
<u>WinHitIvlRight</u>	rechte Seite des <u>Intervalls</u> eines <u>Gantt</u> -Graphen
<u>WinHitAxis</u>	<u>Achse</u> eines <u>Gantt</u> -Graphen
<u>WinHitTreeNode</u>	<u>Knoten</u> -Objekt eines <u>TreeView</u>
<u>WinHitDocViewPage</u>	Seite in einem <u>DocView</u> -Objekt

## Kontakt

<u>WinHitGroupColumn</u>	Spalte eines <u>RecView</u> -Objektes
<u>WinHitGroup</u>	Gruppe eines <u>RecView</u> -Objektes
<u>WinHitCanvasView</u>	Ausgabebereich eines <u>Canvas</u> -Objektes
<u>WinHitCanvasGraphic</u>	<u>CanvasGraphic</u> -Objekt eines <u>Canvas</u> -Objektes aItem

Der Übergabeparameter enthält in Abhängigkeit von dem in aHitTest übergebenen Wert den Deskriptor des angeklickten Objekts:

<u>aHitTest</u>	aItem
<u>WinHitNone</u>	0
<u>WinHitListView</u>	Deskriptor der Spalte, die angeklickt wurde
<u>WinHitListHeader</u>	Deskriptor der Spalte, deren Spaltenkopf angeklickt wurde
<u>WinHitIvl</u>	Deskriptor des angeklickten Interval-Objektes
<u>WinHitIvlLeft</u>	Deskriptor des Interval-Objektes, dessen linke Begrenzung angeklickt wurde
<u>WinHitIvlRight</u>	Deskriptor des Interval-Objektes, dessen rechte Begrenzung angeklickt wurde
<u>WinHitAxis</u>	Deskriptor der angeklickten Achse
<u>WinHitTreeNode</u>	Deskriptor des angeklickten Knoten-Objektes
<u>WinHitDocViewPage</u>	0
<u>WinHitGroupColumn</u>	Deskriptor der Spalte, die angeklickt wurde
<u>WinHitGroup</u>	Deskriptor der Gruppe, die angeklickt wurde
<u>WinHitCanvasView</u>	0
<u>WinHitCanvasGraphic</u>	Deskriptor des angeklickten <u>CanvasGraphic</u> -Objektes im <u>Canvas</u> -Objekt

Wurde das Ereignis bei einem RecView-Objekt mit Hittest WinHitGroup ausgelöst, sind die Eigenschaften SelectorItem und SelectorSubItem entsprechend gesetzt, so dass auf die Eigenschaften des entsprechenden Items zugegriffen werden kann. Die Eigenschaften können nur gelesen werden.

### aID

Enthält aItem einen gültigen Deskriptor auf eine Spalte oder Gruppe, wird in aID entweder die Datensatz-ID des angeklickten Datensatzes (bei den Objekten RecList, RecListPopup und RecView) oder die Zeilennummer (bei den Objekten DataList und DataListPopup) übergeben.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

Enthält aItem einen gültigen Deskriptor auf ein Intervall, enthält aID den Wert, der in der Eigenschaft ID des Intervalls enthalten ist.

Enthält aItem einen gültigen Deskriptor auf eine Achse, wird in aID die Nummer der Skala-Zelle ( $>= 0$ ) übergeben. Liegt der Punkt, der zum Auslösen des Ereignisses führte, im Titel der Achse, wird in aID der Wert -1 übergeben.

## **Kontakt**

**Wurde in einem DocView-Objekt eine Seite angeklickt, wird die Seitennummer in diesem Übergabeparameter übergeben.**

### **Resultat**

**Der Rückgabewert der Funktion wird nicht ausgewertet.**

## Kontakt

### EvtMenuInitPopup

Initialisierung eines Menüfensters

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen

aMenuItem handle	Auslösender Menüeintrag
	Wird nicht

Resultat	logic	ausgewertet
Siehe		<u>Liste, Objekte,</u>

Ereignisbefehle, Beispiel

Bevor das Menüfenster angezeigt wird, wird dieses Ereignis ausgelöst, um **Menüeinträge** zu initialisieren. Menüeinträge können dadurch beispielsweise aktiviert oder deaktiviert werden.

**Definition des Funktionskopfes:**

```
sub EvtMenuInitPopup( aEvt : event; // Ereignis aMenuItem : handle
```

**aEvt**

In diesem Parameter vom Typ **event** wird unter anderem der Deskriptor des auslösenden Objektes übergeben. Dies ist der Deskriptor des Objekts, in dem die Ereignisfunktion eingetragen wurde.

Ausgehend von diesem Objekt kann der Deskriptor des aufgerufenen Menüs mit dem Befehl **WinInfo()** mit den Parametern **WinMenu** oder **WinContextMenu** ermittelt werden. So kann auch bestimmt werden, welches Menü aufgerufen wurde.

**aMenuItem**

Als Objekt wird der Menüeintrag übergeben, der angeklickt wurde, um das Menüfenster zu öffnen. Dies ist entweder ein Menüeintrag in der Menüzeile oder ein Menüeintrag, der ein weiteres Menüfenster öffnet.

Beim Aufruf eines Kontextmenüs wird sofort ein Menüfenster geöffnet. In diesem Fall wird als Objekt 0 übergeben.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Beispiele:**

```
sub EvtMenuInitPopup( aEvt : event; // Ereignis aMenuItem : handle; // auslösender Menü
```

**Unterscheidung des aufgerufenen Menüs:**

```
switch (true){ case (aEvt:Obj->WinInfo(_WinMenu) > 0) : // Menü oder Popup-Menü wurde auf
```

Wird der Deskriptor des Menüs noch im weiteren Verlauf der Funktion benötigt, kann auch folgendes Konstrukt verwendet werden:

## Kontakt

```
tMenu # aEvt:Obj->WinInfo(_WinContextMenu);if (tMenu > 0){ // Contextmenu ...}else{ tMenu # aE
```

## Kontakt

### EvtMenuPopup

Menü einer Schaltfläche aufrufen

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen Wird nicht ausgewertet

Siehe

ausgewertet

Liste, Objekte,

Ereignisbefehle

Die angegebene Prozedur wird ausgeführt, wenn bei einem MenuButton-Objekt die Schaltfläche zum Öffnen des Menüs oder die Tastenkombination  bzw.  gedrückt wird. Das Ereignis wird vor dem Laden des in der Eigenschaft MenuName eingetragenen Menüs aufgerufen. In dem Ereignis kann der Inhalt der Eigenschaft geändert werden.

Definition des Funktionskopfes:

```
sub EvtMenuPopup( aEvt : event; // Ereignis): logic;
```

aEvt

Im Parameter aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes übergeben. Das Objekt entspricht demjenigen, in dem die Ereignisfunktion eingetragen wurde.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Key (Tastatur-Ereignisse)**

Liste sortiert

nach

Gruppen,

Siehe Alphabetische

Liste aller

Ereignisse

- EvtKeyItem

## Kontakt

### EvtKeyItem

#### Tastaturereignis

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aKey	int	Taste
aID	int / bigint	Datensatz-ID bei RecList-Objekten
Resultat	logic	Taste an Objekt weiterleiten?  Siehe <a href="#">Liste, Objekte</a> , <a href="#">Ereignisbefehle</a>

Das Ereignis wird ausgelöst, wenn eine Taste gedrückt wurde.



Das Ereignis wird bei nur in Kombination mit , , , , und ausgelöst.

Für die Pfeiltasten , , und wird das Ereignis nur bei den Objekten [RecList](#), [DataList](#) und [Picture](#) ausgelöst. In diesem Fall wird in dem Parameter [aID](#) der Wert 0 übergeben.

Bei den Objekten [Frame](#) und [AppFrame](#) wird das Ereignis unter folgenden Bedingungen nicht ausgelöst:

1. Ein untergeordnetes Objekt hat den Eingabefokus, bei diesem Objekt ist das Ereignis EvtKeyItem gesetzt und die Ereignis-Prozedur gibt [false](#) zurück.
2. Ein untergeordnetes Objekt hat den Eingabefokus und bearbeitet die Taste selber (z. B. beim Drücken der -Taste, wodurch zum nächsten Objekt gesprungen wird oder die Betätigung von , wenn im Frame ein Default-Button aktiv ist).
3. Die Eigenschaft [MenuKey](#) bei einem [MenuItem](#) oder [Toolbar-Button](#) ist gesetzt und das Objekt ist aktiviert ([Disabled](#) = [false](#)) und sichtbar ([Visible](#) = [true](#)).

Definition des Funktionskopfes:

```
sub EvtKeyItem( aEvt : event; // Ereignis aKey : int;
```

Alternativer Funktionskopf:

```
sub EvtKeyItem( aEvt : event; // Ereignis aKey : int;
```

aEvt

In diesem Parameter vom Typ [event](#) wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aKey

aKey liefert die gedrückte Taste. Die Werte entsprechen den folgenden Konstanten. Die Werte können mit [WinKeyShift](#), [WinKeyCtrl](#) und [WinKeyAlt](#) kombiniert sein,

## Kontakt

wenn entsprechende Tastenkombinationen gedrückt wurden.

Konstante	Taste	Bemerkung	
_WinKeySpace		nicht in Kombination mit	
_WinKeyReturn		nicht in Kombination mit <u>EmulateKeys</u> gesetzt ist	/ nicht wenn
_WinKeyEsc		nicht in Kombination mit	oder
_WinKeyTab		nicht in Kombination mit	
_WinKeyInsert		nicht in Kombination mit	
_WinKeyDelete		nicht in Kombination mit	
_WinKeyBackspace		nicht in Kombination mit	
_WinKeyHome		nicht in Kombination mit	
_WinKeyEnd		nicht in Kombination mit	
_WinKeyPageUp		nicht in Kombination mit	
_WinKeyPageDown		nicht in Kombination mit	
_WinKeyUp		nicht wenn <u>EmulateKeys</u> gesetzt ist	
_WinKeyDown		nicht wenn <u>EmulateKeys</u> gesetzt ist	
_WinKeyLeft			
_WinKeyRight			
_WinKeyF1 ...	...	nicht in Kombination mit	
_WinKeyF12			
_WinKeyA ... _WinKeyZ	...	nicht in Kombination mit	
_WinKey0 ... _WinKey9	...	nicht in Kombination mit	

Die folgenden Tasten lösen das Ereignis nur aus, wenn in der Eigenschaft Flags des Application-Objekts die Ausprägung WinAppCalcKeySupport gesetzt ist:

Konstante	Taste
_WinKeyAdd	
_WinKeySubtract	
_WinKeyMultiply	
_WinKeyDivide	
_WinKeyDecimal	
_WinKeyDot	

Die folgenden Tasten lösen das Ereignis nur aus, wenn in der Eigenschaft FlagsExt des Application-Objekts die Ausprägung WinAppExtFlowKeySupport gesetzt ist:

Konstante	Taste	Bemerkung
_WinKeyTab		
_WinKeyReturn		nur benötigt, wenn <u>EmulateKeys</u> gesetzt ist

## Kontakt

<u>_WinKeyUp</u>		nur benötigt, wenn <u>EmulateKeys</u> gesetzt ist
<u>_WinKeyDown</u>		nur benötigt, wenn <u>EmulateKeys</u> gesetzt ist

Die folgenden Tasten lösen das Ereignis nur aus, wenn in der Eigenschaft FlagsExt des Application-Objekts die Ausprägung WinAppExtUmlautKeySupport gesetzt ist:

Konstante      Taste

\_WinKeyAuml  
\_WinKeyOuml  
\_WinKeyUuml  
\_WinKeySZlig

aID

In Abhängigkeit vom auslösenden Objekt werden hier unterschiedliche Informationen übergeben:

Objekt	Inhalt
<u>RecList /</u> <u>RecListPopup</u>	<u>Datensatz-ID</u> des aktuellen Datensatzes
<u>TreeView</u>	Deskriptor des selektierten <u>TreeNode</u> oder <b>0</b>
<u>Frame</u> oder <u>AppFrame</u>	Deskriptor des Fokus-Objekts
<u>StoList /</u> <u>StoListPopup</u>	0. Der selektierte Eintrag kann über die Eigenschaft <u>Current</u> ermittelt werden.
<u>WebNavigator</u>	0.

Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

Resultat

Unabhängig vom auslösenden Objekt entscheidet der Rückgabewert darüber, ob die auslösende Taste an das Objekt weitergeleitet wird (true) oder nicht (false). Soll zum Beispiel in einem RecList-Objekt das Blättern mit den Tasten

einem Eingabe-Objekt das Arbeiten mit der Zwischenablage ( + ) möglich sein, muss true zurückgegeben werden.



Beispiele:

```
sub EvtKeyItem(  aEvt      : event;    // Ereignis  aKey      : int;      // Taste   aID
```

**Placement (Positionierungs-Ereignisse)**

Liste sortiert

nach

Gruppen,

Alphabetische

Liste aller

Siehe

- EvtPosChanged

## Kontakt

### EvtPosChanged

Änderung der Position/Größe des Objekts

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aRect</u>	<u>rect</u>	Größe des Fensters
<u>aClientSize</u>	<u>point</u>	Größe des Client-Bereichs
<u>aFlags</u>	<u>int</u>	Aktion
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>

Das Ereignis wird ausgelöst, wenn das Objekt in der Position oder Größe verändert wird. Dieses Ereignis dient der Anpassung bzw. Ausrichtung von Unterobjekten des auslösenden Objekts.



Befindet sich in einem Frame-Objekt ein FrameClient-Objekt, wird das Ereignis bis einschließlich zur Version 5.7.08 nicht beim Frame, sondern nur beim FrameClient ausgelöst. Um das Ereignis beim Frame-Objekt auszulösen, muss das Ereignis beim Frame-Objekt gesetzt, und der Dialog nach der 5.7.08 erneut gespeichert worden sein. Dynamisch erzeugte Dialoge müssen nicht gespeichert werden.

Definition des Funktionskopfs:

```
sub EvtPosChanged( aEvt : event;      // Ereignis    aRect : rect;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aRect

In aRect werden die neuen Ausmaße des Objekts angegeben. Dies sind die Außenmaße des Objekts (bei einem Frame zum Beispiel inklusive Rand, Titel usw.). Wird das Fenster minimiert stehen in aRect:left und aRect:top -32000.

aClientSize

In aClientSize wird die Größe des Innenbereichs des Objektes übergeben. Bei Frame-Objekten ist der Innenbereich die Größe des Fensters abzüglich der Titel- und Menüleiste, sowie des Randes. Die neue Breite und Höhe sind die entscheidenden Werte zur Positionierung und Größenveränderung von untergeordneten Objekten. Sichtbare Objekte müssen zwischen den Werten (0,0) und (aClientSize:x,aClientSize:y) liegen.

aFlags

Hier wird die Aktion angegeben, welche durchgeführt wurde:

## Kontakt

<u>WinPosSized</u>	Objekt wurde in der Größe geändert
<u>WinPosMoved</u>	Objekt wurde verschoben
<u>WinPosMinimized</u>	Objekt wurde minimiert
<u>WinPosMaximized</u>	Objekt wurde maximiert

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

 Eine Veränderung des auslösenden Objektes darf nicht in dieser Funktion erfolgen, da es sonst zu Rekursionen kommt.

### Beispiel:

```
sub EvtPosChanged( aEvt : event; // Ereignis aRect : rect; // Größe des Fensters
```

**System (System-Ereignisse)**

Liste sortiert

nach

Gruppen,

Siehe

Alphabetische

Liste aller

- EvtCtxEvent
- EvtEndSession
- EvtFsiMonitor
- EvtJob
- EvtSocket
- EvtTapi
- EvtTimer
- EvtUser

## Kontakt

### EvtCtxEvent

Auslösung eines COM-Ereignisses

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aEventID	int	Ereignisnummer
aComArguments	handle	COM-Argument-Objekt Wird nicht

Resultat                    logic    ausgewertet  
                              Liste, Objekte,

Siehe

#### Ereignisbefehle

Dieses Ereignis wird dann aufgerufen, wenn ein zuvor bei dem CtxDocEdit-Objekt mit ComEvtProcessSet() registriertes Ereignis ausgelöst wurde. Im Argument aEventID wird die Nummer des Ereignisses übergeben.

Definition des Funktionskopfes:

```
sub EvtCtxEvent( aEvt : event; // Ereignis aEventID : int;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aEventID

Dieser Parameter enthält die Nummer des ausgelösten COM-Ereignisses des Objektes.

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

aComArguments

In diesem Parameter wird der Deskriptor auf ein COM-Argument-Objekt übergeben. Es dient zur Übergabe von Ereignis-Argumenten von COM nach CONZEPT 16 und zur Argumentrückgabe von CONZEPT 16 nach COM. Die Argumente können mit ComArgGet() ausgelesen und mit ComArgSet() geschrieben werden.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel:

```
sub EvtCtxEvent( aEvt : event; // Ereignis aEventID : int;
```

## Kontakt

### EvtEndSession

#### Beenden der Windows-Sitzung

Name	Typ	Beschreibung
aEvt	event	Ereignisinformation

**Resultat logic**

ausgewertet

Liste, Objekte,

Siehe Ereignisbefehle,  
WinShutdownBlock()

Die angegebene Prozedur wird ausgeführt, wenn die Windows-Sitzung beendet wird.  
Dies kann durch Abmelden, Herunterfahren oder Neu starten ausgelöst werden.

An dieser Stelle können nicht gespeicherte Daten gespeichert werden.



Mit diesem Ereignis kann das Abmelden nicht abgebrochen werden. Nach standardmäßig 5 Sekunden erhält der Benutzer eine Liste der Programme, die noch nicht beendet sind und kann hier dennoch alle Programme ohne weitere Rückfrage im Programm beenden. Je nach Betriebssystemversion werden Meldungsfenster erst angezeigt, wenn der Abmeldevorgang durch den Benutzer abgebrochen wird. Daher sollte auf Meldungsfenster nach Möglichkeit verzichtet werden.

Um das Abmelden des Benutzers zu verhindern, kann WinShutdownBlock() verwendet werden. Er sollte nur dann verwendet werden, wenn eine Operation in Verarbeitung ist, die abgeschlossen werden muss.



Das Ereignis wird nicht ausgelöst, wenn der Client mittels Task-Manager beendet wird.

**Definition des Funktionskopfs:**

```
sub EvtEndSession( aEvt : event; // Ereignis) : logic;
```

**aEvt**

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Beispiel:**

```
sub EvtEndSession( aEvt : event; // Ereignis) : logic // Wird nicht au
```

## Kontakt

### **EvtFsiMonitor**

#### Überwachen von externen Verzeichnissen

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
		<b>Aktion</b>
		<u>FsiMonActionCreate</u>
		Eine Datei wurde angelegt.
		<u>FsiMonActionDelete</u>
		Eine Datei wurde gelöscht.
<u>aAction</u>	<u>int</u>	<u>FsiMonActionModify</u>
		Eine Datei wurde geändert.
		<u>FsiMonActionRename</u>
		Eine Datei wurde umbenannt.
<u>aFileName</u>	<u>alpha</u>	Pfad- und Dateiname
<u>aFileAttrib</u>	<u>int</u>	Attribute der Datei
<u>aFileSize</u>	<u>bigint</u>	Dateigröße
<u>aFileCT</u>	<u>caltime</u>	Datum und Uhrzeit der letzten Änderung
<u>aFileNameOld</u>	<u>alpha</u>	Alter Pfad- und Dateiname, wenn Datei umbenannt wurde
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
		<u>Liste, Objekte, FsiMonitorOpen(),</u>
Siehe		<u>Ereignisbefehle, Verzeichnissüberwachung (Blog)</u>

Dieses Ereignis wird ausgelöst, wenn in einem überwachten Verzeichnis eine Datei angelegt, gelöscht, kopiert oder verändert wird. Das zu überwachende Verzeichnis muss zuvor mit dem Befehl FsiMonitorAdd() angegeben werden.

#### Definition des Funktionskopfes:

```
sub EvtFsiMonitor( aEvt : event; // Ereignis aAction : int;
```

#### aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

#### aAction

In diesem Parameter wird die Aktion übergeben, die zu einer Änderung im Verzeichnis geführt hat. Der Übergabeparameter kann mit folgenden Konstanten verglichen werden:

#### FsiMonActionCreate

## Kontakt

In dem überwachten Verzeichnis wurde eine neue Datei angelegt. Der Name der Datei ist im Parameter aFileName angegeben.

FsiMonActionDelete

In dem überwachten Verzeichns wurde eine Datei gelöscht. Der Name der Datei ist im Parameter aFileName angegeben.

FsiMonActionModify

In dem überwachten Verzeichnis wurde eine Datei verändert. Der Name der Datei ist im Parameter aFileName angegeben.

FsiMonActionRename In dem überwachten Verzeichnis wurde eine Datei umbenannt. Der neue Name der Datei ist im Parameter aFileName angegeben. In aFileNameOld steht der vorherige Name der Datei.



Die stattgefundene Aktion in dem Verzeichnis wird vom Betriebssystem ermittelt. Je nach Programm können unterschiedliche Aktionen zum Beispiel bei der Änderung von einer Datei ermittelt werden. Schreibt ein Programm die Änderungen direkt in die Datei, wird FsiMonActionModify übergeben. Wird von dem Programm die alte Datei gelöscht und eine neue Datei angelegt, wird FsiMonActionCreate übergeben.

**aFileName**

In diesem Parameter wird der Name der Datei, welche die Aktion in aAction betrifft, übergeben.

**aFileAttrib**

Dies sind die Datei-Attribute der neuen oder veränderten Datei. Die Datei-Attribute können mit folgenden Konstanten verglichen werden:

FsiAttrArchive Archiv-Attribut ist gesetzt.

FsiAttrDir Verzeichnis-Attribut ist gesetzt.

FsiAttrExec Die Datei / Das Verzeichnis ist ausführbar.

FsiAttrHidden Die Datei / Das Verzeichnis ist versteckt.

FsiAttrRead Die Datei kann nur gelesen werden.

FsiAttrSystem System-Attribut ist gesetzt.

**aFileSize**

In diesem Parameter wird die Größe der Datei in Byte übergeben.

**aFileCT**

Hier wird das Datum und die Uhrzeit der letzten Änderung der Datei angegeben.

**aFileNameOld**

Dieser Parameter ist nur bei der Aktion FsiMonActionRename gesetzt und enthält den Namen der Datei vor der Umbenennung.

**Resultat**

## Kontakt

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel:

```
sub EvtFsiMonitor( aEvt : event; // Ereignis aAction : int; // D
```

## Kontakt

### EvtJob Job-Ereignis

Name	Typ	Beschreibung
<u>aEvt</u>	Ereignisinformationen	<u>JobControl</u> -Objekt
<u>aJobControl handle</u>	oder <u>Job</u> -Objekt Wird nicht	

Resultat	logic	ausgewertet <u>Liste, Objekte,</u>
Siehe		<u>Ereignisbefehle, JobEvent()</u> <u>JobOpen() JobControl()</u>

Die angegebene Prozedur wird ausgeführt, wenn in einer parallel ausgeführten

Prozedur die Funktion JobEvent() aufgerufen wird.

Dabei sind zwei verschiedene Szenarien möglich:

1. Der Aufruf von JobEvent() erfolgt in einem Job, das Ereignis wird dann in dem Dialog ausgelöst, welcher auf Seite des Jobstarters bei JobOpen() angegeben wurde.
2. Der Aufruf von JobEvent() erfolgt auf der Seite des Jobstarters, das Ereignis wird dann in dem Dialog ausgelöst, welcher auf Seite des Jobs mittels JobControl() und der Option JobEventReceiver angegeben wurde.

Definition des Funktionskopfes:

```
sub EvtJob( aEvt : event; // Ereignis aJobHdl : handle; // Jo
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aJobControl

Deskriptor des JobControl- oder des Job-Objektes, je nachdem ob der Ereignis auf Seite des Jobstarters (JobControl-Objekt) oder auf Seite des Jobs (Job-Objekt) ausgelöst wird. Es ist zu beachten, dass die beiden Objekttypen teilweise unterschiedliche Eigenschaften haben.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel

```
sub EvtJob( aEvt : event; // Ereignis aJobHdl : handle; // Jo
```

## Kontakt

### EvtSocket

#### Kontakt über einen passiven Socket

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aHandle</u>	<u>handle</u>	Socket-Deskriptor
<u>aSubType int</u>		Untertyp des
		Ereignisses
		Wird nicht
<u>Resultat</u>	<u>logic</u>	<u>ausgewertet</u>
Siehe		<u>Liste, Objekte,</u>
		<u>Ereignisbefehle</u>

Die angegebene Funktion wird ausgeführt, wenn über eine passive Socketverbindung eine Verbindung hergestellt wird.

Definition des Funktionskopfs:

```
sub EvtSocket( aEvt : event; // Ereignis aHandle : handle; //
```

aEvt

In aEvt wird ein Parameter vom Typ event übergeben. In diesem Parameter wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aHandle

In aHandle steht der Socket-Deskriptor der neuen Verbindung. Auf diesem Socket sind alle Operationen möglich.

aSubType

Der Untertyp enthält SckEvtConnect. Wenn der Socket nicht mehr benötigt wird, muss er mit SckClose() wieder geschlossen werden. Der Schreib-/Lese-Timeout des Sockets kann erst nach dem Verbindungsauftbau mit SckInfo(..., SckTimeout,...) gesetzt werden.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

## Kontakt

### EvtTapi

#### Ereignis bei Telefonanrufen

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aTapiDevice</u>	<u>handle</u>	Deskriptor des Tapi-Devices
<u>aCallID</u>	<u>int</u>	Call-ID
<u>aCallState</u>	<u>int</u>	Status des Anrufs
<u>aCallTime</u>	<u>caltime</u>	Datum und Uhrzeit des Anrufes
<u>aCallerID</u>	<u>alpha</u>	Rufnummer des Anrufers
<u>aCalledID</u>	<u>alpha</u>	Rufnummer des Anschlusses, der angerufen wird
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte, TapiListen()</u> , <u>Ereignisbefehle, Aus- und Eingehende Telefonate (Blog),</u> <u>Ereignisreihenfolge (Blog),</u> <u>Monitoring (Blog)</u>

Die angegebene Prozedur wird ausgeführt, wenn die TAPI-Schnittstelle ein Ereignis für einen eingehenden oder ausgehenden Anruf signalisiert. Für das Gerät muss zuvor mit dem Befehl TapiListen() die Überwachung aktiviert worden sein.

Definition des Funktionskopfes:

```
sub EvtTapi(  aEvt           : event;      // Ereignis  aTapiDevice        : handle;    //
```

**aEvt**

In **aEvt** wird ein Parameter vom Typ event übergeben. In diesem Parameter wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aTapiDevice**

In diesem Parameter wird der Deskriptor des TAPI-Gerätes, dass das Ereignis auslöste, übergeben. Dabei handelt es sich um den Deskriptor, der durch TapiDial() oder TapiListen() zurückgeliefert wurde.

**aCallID**

Der hier übergebene Wert kennzeichnet die eingehenden oder ausgehenden Anruf eindeutig. Die Call-ID wird durch den Telefon-Service-Provider generiert. Jeder Anruf hat seine eigene ID.

**aCallState**

## Kontakt

Dieser Parameter gibt den Zustand des Anrufs an. Der übergebene Wert kann mit folgenden Konstanten verglichen werden:

Konstante	Eingehender	Ausgehender	Bemerkung
	Anruf	Anruf	
<u><a href="#">TapiCallStateInfo</a></u>			Die Informationen <u>aCallerID</u> und <u>aCalledID</u> haben sich geändert.
<u><a href="#">TapiCallStateOffer</a></u>			Ein neu einkommender Anruf wird signalisiert. <u>aCallerID</u> und <u>aCalledID</u> enthalten, sofern vom TAPI-Treiber unterstützt, entsprechende Anruf- und Anrufer-Informationen.
<u><a href="#">TapiCallStateProceeding</a></u>			Der Zustand kann nur bei asynchronem <u>TapiDial()</u> auftreten. Der Wählvorgang wurde abgeschlossen und wird vom Telefonnetz weitervermittelt. Der Zustand tritt nach dem Wählen jedoch vor <u>TapiCallStateRingBack</u> , <u>TapiCallStateBusy</u> , <u>TapiCallStateConnected</u> oder <u>TapiCallStateDisconnected</u> auf.
<u><a href="#">TapiCallStateRingBack</a></u>			Der Zustand kann nur bei asynchronem <u>TapiDial()</u> auftreten. Die Zieladresse wurde erreicht.
<u><a href="#">TapiCallStateBusy</a></u>			Der Zustand kann nur bei asynchronem <u>TapiDial()</u> auftreten. Der Anruf bekommt ein Besetzt-Zeichen zurückgeliefert.
<u><a href="#">TapiCallStateConnected</a></u>			Die Verbindung wurde hergestellt.
<u><a href="#">TapiCallStateDisconnected</a></u>			Die entfernte Station hat den Anruf getrennt.
<u><a href="#">TapiCallStateOnHold</a></u>			Der Anruf wird gehalten. Dadurch kann ein weiterer Anruf entgegen genommen werden.
<u><a href="#">TapiCallStateIdle</a></u>			Dies ist der finale Zustand, der das Ende der

## Kontakt

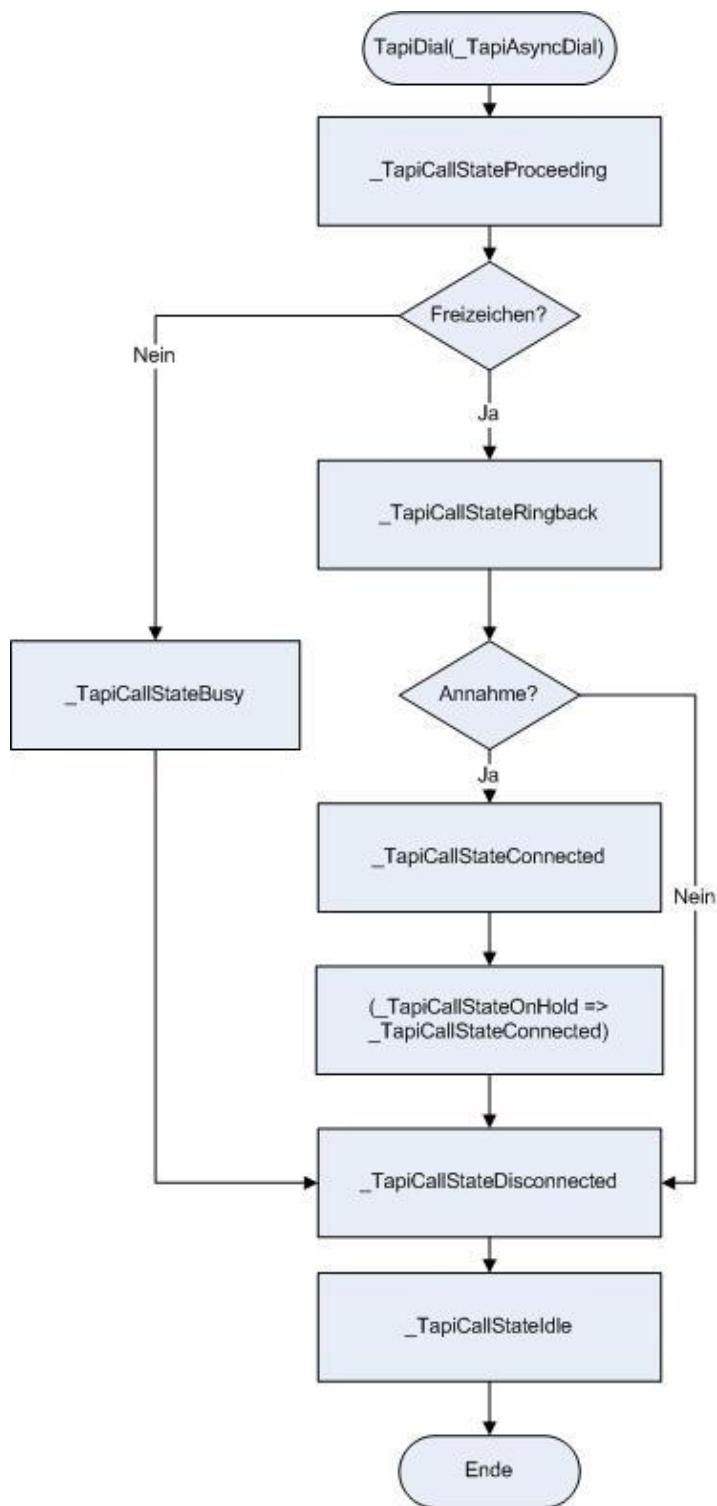
### TapiCallStateUnknown

Verbindung kennzeichnet.  
Die in aCallID angegebene  
Nummer ist nicht mehr  
gültig.

Wird übergeben, wenn der  
aktuelle Verbindungsstatus  
nicht ermittelt werden  
kann. Diese Konstante wird  
auch übergeben, wenn die  
Überwachung und ein  
TapiDial() auf demselben  
TAPI-Gerät durchgeführt  
werden.

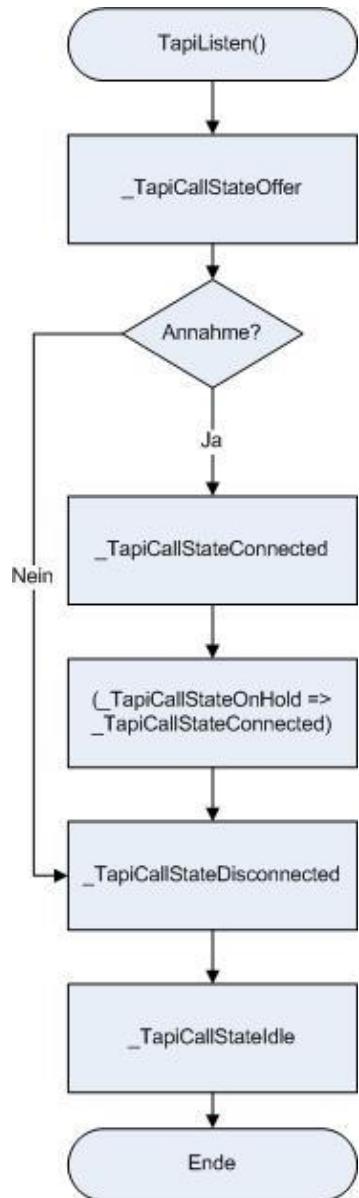
Bei ausgehenden Anrufen wird das EvtTapi mit CallStates in der folgenden  
Reihenfolge aufgerufen:

## Kontakt



Bei eingehenden Anrufen ist es diese Reihenfolge:

## Kontakt



### aCallTime

Hier wird das Datum und die Uhrzeit des Anrufes übergeben. Der Aufruf des Ereignisses kann sich durch die Verarbeitung anderer Funktionen verzögern. Der Übergabeparameter gibt den tatsächlichen Zeitpunkt des Ereignisses an. Der Übergabewert kann mit den bei caltime angegebenen Methoden in die einzelnen Bestandteile zerlegt werden.

### aCallerID und aCalledID

In diese Zeichenketten werden die Rufnummer des Anrufers und des Angerufenen übergeben, die die TAPI-Schnittstelle vom TAPI Service Provider bekommen hat. Die Informationen können auch leer sein, wenn der Provider sie nicht ermitteln konnte. Bei einem eingehenden Anruf sind diese Informationen normalerweise vorhanden, wenn der Zustand TapiCallStateOffer oder TapiCallStateConnected ist.

## Kontakt

Bei einem ausgehendem Anruf sind diese Informationen normalerweise vorhanden, wenn der Zustand TapiCallStateProceeding oder TapiCallStateConnected ist.

Der Aufbau der Zeichenketten und welcher Wert zurückgegeben wird, wenn die Rufnummer unbekannt ist, hängt vom verwendeten Gerätetreiber ab.

## Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

## Beispiel

```
sub EvtTapi( aEvt          : event;      // Ereignis  aTapiDevice : handle;    // Tapi Device  aCal
```

## Kontakt

### EvtTimer

#### Zeitgesteuertes Ereignis

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen

#### aTimerID handle Timer-Deskriptor

<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe	<u>SysTimerCreate()</u> , <u>Ereignisbefehle</u>	

Die angegebene Prozedur wird ausgeführt wenn ein Timer-Ereignis über die Funktion SysTimerCreate() gestartet wurde.

Definition des Funktionskopfs:

```
sub EvtTimer(    aEvt                      : event; // Ereignis  aTimerID          : handle; // Tim
```

aEvt

In aEvt wird ein Parameter vom Typ event übergeben. In diesem Parameter wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aTimerID

Hier wird der Deskriptor des Timer-Ereignisses übergeben. Der übergebene Wert kann mit dem Rückgabewert des Befehls SysTimerCreate() verglichen werden, um den Timer zu identifizieren.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

## Kontakt

### EvtUser

#### Benutzerdefiniertes Ereignis

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aEventID</u>	<u>word</u>	Benutzerdefinierte
<u>aArgument1</u> int		Ereignis-Nummer Numerisches

Argument

#### aArgument2 alpha(8192) Alphanumerisches Argument

Wird nicht

<u>Resultat</u>	<u>logic</u>	ausgewertet
Siehe		<u>Liste</u> , <u>Objekte</u> , <u>WinUserEvent()</u> ,

Ereignisbefehle

Die angegebene Funktion wird ausgeführt, wenn ein Ereignis vom Programmierer ausgelöst wurde. Das Auslösen erfolgt über den Befehl WinUserEvent().

Definition des Funktionskopfs:

```
sub EvtUser( aEvt : event;      // Ereignis aEventID : word;      // B
```

aEvt

In aEvt wird ein Parameter vom Typ event übergeben. In diesem Parameter wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aEventID, aArgument1 und aArgument2

In diesen Parametern werden die Argumente übergeben, die bei der Anweisung WinUserEvent() übergeben wurden.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Dieses Ereignis dient dazu Funktionen auszuführen, die nicht in dem ursprünglichen Ereignis ausgeführt werden sollten. Zum Beispiel sollte ein Dialog nicht in dem Ereignis EvtFocusTerm aufgerufen werden, da nach dem Beenden des Fensters das Eingabe-Objekt wieder den Fokus bekommt. Innerhalb des EvtFocusTerm kann ein EvtUser ausgelöst werden, in dem dann der Dialog gestartet wird.

## Special (Objektspezifische Ereignisse)

Liste sortiert

nach

Gruppen,

Siehe

Alphabetische

Liste aller

- EvtAdviseDDE
- EvtAttachState
- EvtChanged
- EvtChangedChild
- EvtChangedDesign
- EvtClicked
- EvtCroNavigate
- EvtDbFldUpdate
- EvtHelpTip
- EvtHyphenate
- EvtLstDataInit
- EvtLstEditActivate
- EvtLstEditCommit
- EvtLstEditEndGroup
- EvtLstEditEndItem
- EvtLstEditFinished
- EvtLstEditStart
- EvtLstEditStartGroup
- EvtLstEditStartItem
- EvtLstGroupArrange
- EvtLstGroupInit
- EvtLstRecControl
- EvtLstSelect
- EvtLstSelectRange
- EvtLstViewInit
- EvtNodeExpand
- EvtNodeSearch
- EvtNodeSelect
- EvtPageSelect
- EvtReadOnlyChanged

## Kontakt

### EvtAdviseDDE

Daten einer DDE-Verbindung stehen bereit

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aDdeHdl</u>	<u>handle</u>	Deskriptor der DDE-Verbindung
<u>aDdeItemName alpha</u>		Name des Elementes
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte,</u> <b>Ereignisbefehle</b>

Dieses Ereignis wird ausgelöst, wenn von einem DDE-Server Daten an die Applikation gesendet werden. Das Ereignis wird nur ausgelöst, wenn das empfangende Fenster-Objekt (AppFrame-, Frame- oder MdiFrame-Objekt) sichtbar ist und das sendende Kommunikationselement des DDE-Servers für das Senden von Daten zugelassen wurde (DdeCommand(..., DdeAdviseOn, ...)).

Definition des Funktionskopfes:

```
sub EvtAdviseDDE( aEvt : event; // Ereignis aDdeHdl : handle;
```

**aEvt**

In diesem Übergabeparameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aDdeHdl und aDdeItemName**

In aDdeHdl und aDdeItemName werden der Deskriptor des DDE-Kanals und der Name des sendenden Elements des DDE-Servers übergeben. Diese Informationen werden benötigt, um die gesendeten Daten mit dem Befehl DdeCommand() an die Applikation zu übermitteln und mit DdeGetData() auszulesen.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Beispiel:**

```
sub EvtAdviseDDE( aEvt : event; // Ereignis aDdeHdl : handle;
```

## Kontakt

### EvtAttachState

Änderung im Zustand des GroupTile-Objekts

	Name	Typ	Beschreibung
	<u>aEvt</u>	<u>event</u>	Ereignisinformationen
			Ereignis-Modus
			<u>WinModeAttachQuery</u> Zustandsänderung durchführen?
<u>aMode</u>	<u>int</u>		<u>WinModeAttachChanged</u> Zustandsänderung durchgeführt.
			Vorhergehender Zustand
			<u>WinStateAttachNormal</u> <u>GroupTile</u> -Objekt wird normal dargestellt.
<u>aStateOld</u>	<u>int</u>		<u>WinStateAttachMaximized</u> <u>GroupTile</u> -Objekt wird maximiert dargestellt.
			<u>WinStateAttachClosed</u> <u>GroupTile</u> -Objekt ist geschlossen.
			neuer Zustand
			<u>WinStateAttachNormal</u> <u>GroupTile</u> -Objekt soll normal dargestellt.
			<u>WinStateAttachMaximized</u> <u>GroupTile</u> -Objekt soll maximiert dargestellt.
<u>aStateNew</u>	<u>int</u>		<u>WinStateAttachClosed</u> <u>GroupTile</u> -Objekt soll geschlossen werden.
			Grund für die Änderung des Zustands
			<u>WinReasonUserCaption</u> Titelleiste
<u>aReason</u>	<u>int</u>		<u>WinReasonUserTab</u> -Taste
			<u>WinReasonInternProcedure</u> prozedural
			<u>WinReasonInternIndirect</u> indirekt
<u>Resultat</u>	<u>logic</u>		Zustandsänderung durchführen?
Siehe			<u>Liste</u> , <u>Objekte</u> , <u>Ereignisbefehle</u>

Das Ereignis wird aufgerufen, wenn eine Zustandsänderung erfolgen soll, also das GroupTile-Objekt geschlossen, maximiert oder wiederhergestellt werden soll.

Dabei wird das Ereignis zunächst mit der Konstante WinModeAttachQuery in aMode aufgerufen. Durch den Rückgabewert von true wird die Zustandsänderung zugelassen, durch einen Rückgabewert von false wird sie verhindert.

Wurde die Zustandsänderung zugelassen, wird das Ereignis nochmals mit der Konstante WinModeAttachChanged aufgerufen, um zu benachrichtigen, dass die Zustandsänderung durchgeführt wurde.

Definition des Funktionskopfes:

## Kontakt

```
sub EvtAttachState( aEvt : event; // Ereignis aMode : int;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aMode

Der hier übergebene Wert kann mit den folgenden Konstanten verglichen werden:

WinModeAttachQuery Nachfrage, ob Zustandsänderung durchgeführt werden darf.

WinModeAttachChanged Erfolgte Zustandsveränderung.

aStateOld / aStateNew

In diesen Parametern wird übergeben, in welchem Zustand das GroupTile-Objekt sich befindet und welchen Zustand es annehmen soll. Der Übergabewert kann mit folgenden Konstanten verglichen werden:

WinStateAttachNormal GroupTile-Objekt wird normal dargestellt.

WinStateAttachMaximized GroupTile-Objekt wird maximiert dargestellt.

WinStateAttachClosed GroupTile-Objekt ist geschlossen.

aReason

In diesem Parameter wird der Grund der Zustandsänderung übergeben. Er kann mit folgenden Konstanten verglichen werden:

WinReasonUserCaption Benutzerinteraktion mit der Titelleiste

WinReasonUserTab Benutzernavigation durch die  -Taste

WinReasonInternProcedure prozedural ausgelöst (zum Beispiel durch Setzen einer Eigenschaft)

WinReasonInternIndirect indirekt durch die Zustandsveränderung eines anderen GroupTile-Objekts ausgelöst

Resultat

Der Rückgabewert entscheidet, um die Zustandsänderung durchgeführt wird (true) oder nicht (false). Wird in aMode WinModeAttachChanged übergeben, ist der Rückgabewert ohne Bedeutung.

### EvtChanged

Änderung im Eingabe-Objekt

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen Wird nicht

Resultat logic

**ausgewertet**  
Siehe [Liste, Objekte](#),  
[Ereignisbefehle](#)

- [Edit-Objekte](#)

Das Ereignis wird ausgelöst, wenn durch den Benutzer in einem [Edit-Objekt](#) (inklusive des [RtfEdit-Objekts](#)) eine Änderung des Inhaltes vorgenommen wird.

- [DocView](#)

Das Ereignis wird ausgelöst, wenn eine Vergrößerung/Verkleinerung vorgenommen wird.

Das Ereignis wird ebenfalls aufgerufen, wenn eine Änderung der Größendarstellung durch Setzen der Eigenschaft [ZoomFactor](#) oder [PageZoom](#) erfolgt.

- [RtfEdit](#)

Das Ereignis wird zusätzlich ausgelöst, wenn ein bestehender Text durch einen identischen Text ersetzt wird. Gleches gilt, wenn bei einem leeren Text weiter versucht wird, Text zu entfernen.

- [Chromium](#) Wird ausgelöst, nachdem sich die Url des Inhaltes geändert hat. Die Url ist über die Eigenschaft [wpUrl](#) lesbar.

Durch Setzen der Eigenschaft [ChangedTrigger](#) beim übergeordneten [Frame- oder MDI-Frame-Objekt kann das Ereignis auch beim prozeduralen Ändern der \[Caption-Eigenschaft oder beim Übertragen der Feldpuffer in das Objekt ausgelöst werden.\]\(#\)](#)

Wird das Ereignis das erste Mal für das übergebene Objekt ausgelöst, hat die Eigenschaft [Changed](#) den Wert [false](#). Dadurch ist eine Unterscheidung der EvtChanged-Ereignisse möglich.

Definition des Funktionskopfes:

```
sub EvtChanged( aEvt : event; // Ereignis) : logic;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ [event](#). In aEvt wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel:

## Kontakt

```
sub EvtChanged(    aEvt  
                  : event;      // Ereignis) : logic;
```

## Kontakt

### EvtChangedChild

Änderung in untergeordnetem Eingabe-Objekt

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen

aObject handle Auslösendes Objekt

Resultat logic Wird nicht ausgewertet

Siehe [Liste, Objekte, Ereignisbefehle](#)

Dieses Ereignis wird nur ausgelöst, wenn in der Eigenschaft [ChangedTrigger](#) die Ausprägung [WinChgTriggerChild](#) gesetzt ist.

Erfolgt in einem untergeordneten Eingabe-Objekt eine Änderung, wird das Ereignis aufgerufen.

Das Ereignis wird ebenfalls aufgerufen, wenn das entsprechende Eingabe-Objekt über ein Ereignis [EvtChanged](#) verfügt.

Durch Setzen der Eigenschaft [ChangedTrigger](#) kann das Ereignis auch beim prozeduralen Ändern der [Caption](#)-Eigenschaft oder beim Übertragen der Feldpuffer in das Objekt ausgelöst werden.

Wird das Ereignis für das Fenster-Objekt ausgelöst, wird die Eigenschaft [ChangedChild](#) auf den Wert [true](#) gesetzt. Über diese Eigenschaft kann an einer unabhängigen Stelle (zum Beispiel beim [EvtClicked](#) oder [EvtClose](#)) festgestellt werden, ob mindestens ein Eingabe-Objekt verändert wurde. Die Eigenschaft kann jederzeit zurückgesetzt werden.

Definition des Funktionskopfes:

```
sub EvtChangedChild(    aEvt          : event;      // Ereignis   aObject        : handl
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ [event](#). In aEvt wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aObject

aObject enthält den Deskriptor des geänderten Eingabe-Objekts.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel:

```
sub EvtChangedChild(    aEvt          : event;      // Ereignis   aObject        : handl
```

## Kontakt

### EvtChangedDesign

Änderung der Position/Größe des Objekts im Design-Modus

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aAction</u>	<u>int</u>	Durchgeführte Aktion
<u>Resultat</u>	<u>logic</u>	Änderung durchführen?
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>

Das Ereignis wird ausgelöst, wenn das Objekt im Design-Modus (siehe Design) in der Position oder Größe verändert wird.

**Definition des Funktionskops:**

```
sub EvtChangedDesign( aEvt : event; // Ereignis aAction : int
```

**aEvt**

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aAction**

Hier wird die Aktion angegeben, welche durchgeführt wurde. Zur Zeit kann nur die Größe oder die Position des Objekts geändert werden. Diese Aktion wird mit dem Wert \_WinDesignActionMoveSize (1) angegeben.

**Resultat**

Über den Rückgabewert kann bestimmt werden, ob die Änderung an dem Objekt durchgeführt werden soll (true) oder nicht (false).

**Beispiel:**

```
sub EvtChangedDesign( aEvt : event; // Ereignis aAction : int; // Aktion) : logic;
```

## Kontakt

### EvtClicked

Schaltfläche wurde gedrückt

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen

Resultat logic Standardaktion des  
Buttons ausführen?

Siehe Liste, Objekte,

#### Ereignisbefehle

Die angegebene Prozedur wird ausgeführt, wenn die Schaltfläche mit der Maus oder mit der Tastatur ausgewählt wurde. Ist das erste Objekt eines Fensters das den Fokus erhalten kann ein Radiobutton, wird dieser Radiobutton bei Erzeugung des Fensters fokussiert und somit ebenfalls das Ereignis EvtClicked ausgelöst.

Definition des Funktionskopfes:

```
sub EvtClicked( aEvt : event; // Ereignis) : logic;
```

aEvt

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

Resultat

Ist bei einem Button-Objekt die Eigenschaft TypeButton auf WinBtnClose oder WinBtnHelp gesetzt, wird über den Rückgabewert der Funktion entschieden, ob die Aktion ausgeführt wird (true) oder nicht (false).

Beispiel:

```
sub EvtClicked( aEvt : event; // Ereignis) : logic; // Standardaktion d
```

## Kontakt

### EvtDbFldUpdate

Datensatzoperation wurde durchgeführt

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>

Die angegebene Prozedur wird ausgeführt, wenn eine Schaltfläche des RecNavigators gedrückt wurde.

**Definition des Funktionskopfes:**

```
sub EvtDbFldUpdate( aEvt : event; // Ereignis): logic;
```

**aEvt**

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Beispiel:**

```
sub EvtDbFldUpdate( aEvt : event; // Ereignis): logic;
```

## Kontakt

### EvtEditorHelp

#### Hilfe-Ereignis im CodeEdit

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aWord</u>	<b>alpha</b>	Wort, zu dem Hilfe angefordert wird
<u>aCaretColumn</u>	<u>int</u>	Spaltennummer der Schreibmarke
<u>aCaretLine</u>	<u>int</u>	Zeilennummer der Schreibmarke
<u>Resultat</u>	<u>logic</u>	Verarbeitung an CodeEdit weiterleiten <u>Liste, Objekte,</u> <u>Ereignisbefehle,</u>
Siehe		

Dieses Ereignis wird aufgerufen, wenn im CodeEdit die Tastenkombination Strg+F1 betätigt wird. Hierdurch wird die kontextsensitive Hilfe zum Befehl an der Schreibmarke angezeigt. Das Ereignis EvtEditorHelp wird vor der Anzeige ausgelöst, um eine selbst definierte Hilfebehandlung vornehmen zu können. Der Rückgabewert entscheidet darüber, ob die Verarbeitung an das CodeEdit weitergeleitet wird (true) oder nicht (false).

#### Definition des Funktionskopfes:

```
sub EvtEditorHelp( aEvt : event; // Ereignis aWord : al
```

#### aEvt

In aEvt vom Typ event wird unter anderem der Deskriptor des auszulösenden Objekts übergeben.

#### aWord

Enthält das Wort an der Position der Schreibmarke, bzw. das selektierte Wort.

#### aCaretColumn

Enthält die Spaltennummer der Schreibmarke.

#### aCaretLine

Enthält die Zeilennummer der Schreibmarke.

#### Resultat

Wird true zurückgegeben, wird die Verarbeitung nach Durchführung des Ereignisses an das CodeEdit weitergeleitet, sonst nicht.

#### Beispiele

```
sub EvtEditorHelp( aEvt : event; // Ereignis aWord : al
```

## Kontakt

### Objekte mit EvtEditorHelp

Folgende Objekte haben das Ereignis EvtEditorHelp.

EvtEditorHelp,

Liste der

Objekte, Liste

der Ereignisse

- CodeEdit

Siehe

**EvtHelpTip****HelpTip-Ereignis**

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aButton</u>	<u>int</u>	Zusatztasten
<u>aHitTest</u>	<u>int</u>	Hit-test-Code
<u>aItem</u>	<u>handle</u>	Unterobjekt
<u>aID</u>	<u>int / bigint</u>	Objekt-ID
<u>aNameID</u>	<u>alpha</u>	Namens-ID
<u>aText</u>	<u>alpha(8192)</u>	Eigenschaft <u>HelpTip</u> des Objektes (veränderbar)
<u>aJustify</u>	<u>int</u>	Eigenschaft <u>HelpTipJustify</u> des Objektes (veränderbar)
<u>Resultat</u>	<u>logic</u>	HelpTip anzeigen? <u>Liste, Objekte, Ereignisbefehle,</u>
Siehe	<u>Blog</u>	

Dieses Ereignis wird aufgerufen, wenn über ein Objekt mit der Eigenschaft HelpTip gefahren wird. Während der HelpTip angezeigt wird, wird das Ereignis nur ausgelöst, wenn sich ein Unterobjekt des Objektes ändert. Ein Unterobjekt könnte beispielsweise eine Zeile oder Spalte einer RecList sein. Das Ereignis wird innerhalb eines Objektes maximal alle 250 Millisekunden ausgelöst, unabhängig davon, ob ein HelpTip angezeigt wird, oder nicht.



Die Verarbeitung des Ereignisses sollte möglichst kurz sein, um den Programmablauf nicht zu unterbrechen. Weitere Ereignisse werden erst nach der Abarbeitung von EvtHelpTip ausgeführt.

Definition des Funktionskopfes:

```
sub EvtHelpTip( aEvt : event; // Ereignis aButton : int;
```

Alternativer Funktionskopf:

```
sub EvtHelpTip( aEvt : event; // Ereignis aButton : int;
```

aEvt

In aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts übergeben.

aButton

aButton gibt die Taste oder die Tastenkombination an, die beim Auslösen des Ereignisses gedrückt waren:

WinMouseShift -Taste

WinMouseCtrl -Taste

## Kontakt

Die Konstanten können miteinander kombiniert werden.

### aHitTest

In diesem Parameter wird der Objekttyp angegeben, der an der Stelle der Mausposition vorliegt:

<u>WinHitNone</u>	kein Objekt
<u>WinHitLstHeader</u>	Spaltenkopf eines RecList- oder DataList-Objektes oder leerer Bereich des Spaltenkopfes eines RecView-Objektes
<u>WinHitLstView</u>	Spalte eines RecList- oder DataList-, Anzeigebereich eines DocView- oder leerer Bereich eines RecView-Objektes
<u>WinHitCorner</u>	Zwischenraum der Achsen (linke obere Ecke eines Gantt-Graphen)
<u>WinHitGanttView</u>	Ausgabebereich eines Gantt-Graphen
<u>WinHitIvl</u>	Interval-Objekt eines Gantt-Graphen
<u>WinHitIvlBox</u>	Box-Objekt eines Gantt-Graphen ( <u>WinGanttIvlBoxClick</u> muss in <u>GanttFlags</u> gesetzt sein)
<u>WinHitIvlLeft</u>	linke Seite des Intervalls eines Gantt-Graphen
<u>WinHitIvlRight</u>	rechte Seite des Intervalls eines Gantt-Graphen
<u>WinHitAxis</u>	Achse eines Gantt-Graphen
<u>WinHitTreeNode</u>	Knoten-Objekt eines TreeView
<u>WinHitRecNavFirst</u>	"ersten Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavLast</u>	"letzten Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavPrev</u>	"vorigen Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavNext</u>	"nächsten Datensatz lesen" eines RecNavigator-Objektes
<u>WinHitRecNavPgUp</u>	"fünf Datensätze zurück blättern" eines RecNavigator-Objektes
<u>WinHitRecNavPgDn</u>	"fünf Datensätze vor blättern" eines RecNavigator-Objektes
<u>WinHitRecNavNew</u>	"Datensatz einfügen" eines RecNavigator-Objektes
<u>WinHitRecNavDel</u>	"Datensatz löschen" eines RecNavigator-Objektes
<u>WinHitRecNavSave</u>	"Datensatz Speichern" eines RecNavigator-Objektes
<u>WinHitRecNavLock</u>	"Datensatz sperren" eines RecNavigator-Objektes
<u>WinHitStatusbar</u>	Statusbar
<u>WinHitStatusbarButton</u>	Statusbar-Button-Objekt in einer Statusbar
<u>WinHitToolbar</u>	Toolbar
<u>WinHitToolbarButton</u>	Toolbar-Button-Objekt in einer Toolbar
<u>WinHitToolbarRtf</u>	ToolbarRtf
<u>WinHitDocViewPage</u>	Seite eines DocView-Objektes
<u>WinHitPrtPage</u>	Seite eines PrtJobPreview-Objektes
<u>WinHitGroup</u>	Gruppe eines RecView-Objektes
<u>WinHitGroupColumn</u>	Spaltenkopf eines RecView-Objektes
<u>WinHitGroupTileButton</u>	GroupTile-Button-Objekt in einem GroupTile
<u>WinHitCanvasView</u>	Ausgabebereich eines Canvas-Objektes

## Kontakt

### WinHitCanvasGraphic CanvasGraphic eines Canvas-Objektes aItem

Der Übergabeparameter enthält in Abhängigkeit von dem in aHitTest übergebenenden Wert den Deskriptor des Objektes, über dem die Maus liegt:

<u>aHitTest</u>	aItem
<u>WinHitNone</u>	0
<u>WinHitListView</u>	Deskriptor der Spalte
<u>WinHitLstHeader</u>	Deskriptor der Spalte, über deren Spaltenkopf die Maus liegt
<u>WinHitIvl</u>	Deskriptor des Interval-Objektes
<u>WinHitIvlLeft</u>	Deskriptor des Interval-Objektes, über dessen linker Begrenzung die Maus liegt
<u>WinHitIvlRight</u>	Deskriptor des Interval-Objektes, über dessen rechter Begrenzung die Maus liegt
<u>WinHitIvlBox</u>	Deskriptor des Box-Objektes
<u>WinHitGanttView</u>	Die Nummer der Spalte, die Zeile wird in aID übergeben.
<u>WinHitAxis</u>	Deskriptor der Achse
<u>WinHitTreeNode</u>	Deskriptor des Knoten-Objektes
<u>WinHitStatusbar</u>	0
<u>WinHitStatusbarButton</u>	Deskriptor des Statusbar-Button-Objekts
<u>WinHitToolbar</u>	0
<u>WinHitToolbarButton</u>	Deskriptor des Toolbar-Button-Objekts
<u>WinHitToolbarRtf</u>	Deskriptor des untergeordneten Objektes
<u>WinHitDocViewPage</u>	0
<u>WinHitPrtPage</u>	Deskriptor des Objektes auf der Seite, über dem die Maus liegt
<u>WinHitGroup</u>	Deskriptor der Gruppe, über der die Maus liegt
<u>WinHitGroupColumn</u>	Deskriptor der Spalte, über deren Spaltenkopf die Maus liegt
<u>WinHitGroupTileButton</u>	Deskriptor des GroupTile-Button-Objektes
<u>WinHitCanvasView</u>	0
<u>WinHitCanvasGraphic</u>	Deskriptor des CanvasGraphic-Objektes im Canvas-Objekt
Wurde das Ereignis bei einem <u>PrtJobPreview</u> -Objekt ausgelöst, kann der Name des Objekts (aItem->wpName) nur dann ermittelt werden, wenn der Druckjob zuvor mit der Option <u>PrtJobOpenVerbose</u> gespeichert wurde. Das <u>PrtJobPreview</u> -Objekt der <u>Druckvorschau</u> kann mit dem Befehl <u>PrtSearch(..., 'PpvControl')</u> ermittelt werden.	
Wurde das Ereignis bei einem <u>RecView</u> -Objekt mit Hittest <u>WinHitGroup</u> ausgelöst, sind die Eigenschaften <u>SelectorItem</u> und <u>SelectorSubItem</u> entsprechend gesetzt, so dass auf die Eigenschaften des entsprechenden <u>Items</u> zugegriffen werden kann. Die Eigenschaften können nur gelesen werden.	

aID

## Kontakt

Enthält aItem einen gültigen Deskriptor auf eine Spalte oder Gruppe, wird in aID entweder die Datensatz-ID des Datensatzes (bei den Objekten RecList, RecListPopup und RecView) oder die Zeilennummer (bei den Objekten DataList und DataListPopup) der Zeile übergeben, über der die Maus liegt.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

Wurde die Maus innerhalb eines Statusbar-oder Toolbar-Objektes positioniert, ist aID immer 0.

Enthält aItem einen gültigen Deskriptor auf ein Intervall, enthält aID den Wert, der in der Eigenschaft ID des Intervalls enthalten ist.

Enthält aItem einen gültigen Deskriptor auf eine Achse, wird in aID die Nummer der Skala-Zelle ( $\geq 0$ ) übergeben. Liegt der Punkt, der zum Auslösen des Ereignisses führte, im Titel der Achse, wird in aID der Wert -1 übergeben.

Wird in aItem die Nummer einer Spalte eines Gantt-Graphen übergeben, befindet sich in diesem Parameter die Zeile, über der die Maus steht.

Wurde das Ereignis bei einem DocView-Objekt ausgelöst, steht in aID die Nummer der Seite. Existiert an der Mausposition keine Seite, wird 0 übergeben. Das Argument aItem ist immer 0.

Ist das Ereignis bei einem PrtJobPreview eingetragen, wird in aID die Seitennummer anhand der Mausposition übergeben. Ist an der Mausposition keine Seite, wird 0 übergeben. In diesem Fall ist aHitTest = WinHitNone und aItem = 0.

Beim Calendar-Objekt wird in aID das mit CnvID() gewandelte Datum übergeben.  
aItem ist immer 0.

aNameID

In diesem Argument wird die Namensreferenz des Objektes übergeben, über dem die Maus steht. Dieses Argument ist nur bei folgenden Objekten gesetzt:

Objekt	Bedeutung
<u>StoList</u> / <u>StoListPopup</u>	Name der Applikations-Ressource

aText

In aText wird der Wert der Eigenschaft HelpTip des Objektes übergeben. Hat ein Untergeordnetes Objekt einen eigenen HelpTip und befindet sich die Maus über diesem Unterobjekt, wird der HelpTip des Unterobjektes übergeben. Beispielsweise kann jede Spalte einer Liste einen eigenen HelpTip besitzen.

Der anzuzeigende Text kann mit diesem Argument verändert werden. Ist aText beim Beenden des Ereignisses eine leere Zeichenkette, wird kein HelpTip angezeigt.

Setzen dieses Argumentes verändert nicht den Wert der Eigenschaft HelpTip.

## Kontakt

 Unterstützt das Objekt **Unicode-Werte**, werden Umlaute auch im Unicode-Zeichensatz übergeben. Bei der Rückgabe müssen im Ereignis gesetzte Umlaute ebenfalls entsprechend gewandelt werden (siehe Beispiele). Die **Unicode-Unterstützung** des Objektes kann mit **WinInfo( WinUnicode)** ermittelt werden.

### aJustify

In aText wird der Wert der Eigenschaft **HelpTipJustify** des Objektes übergeben. Dieser kann auf folgende Werte gesetzt werden:

- **WinJustLeft** Text linksbündig darstellen
- **WinJustCenter** Text zentriert darstellen
- **WinJustRight** Text rechtsbündig darstellen
- **WinJustParent** Text wird wie beim übergeordneten Objekt angegeben dargestellt

Setzen dieses Argumentes verändert nicht den Wert der Eigenschaft **HelpTipJustify**.

### Resultat

Wird **true** zurückgegeben, wird der Text aus aText angezeigt, sofern dieser keine leere Zeichenkette ist.

### Besonderheiten

 Während dieses Ereignisses ist eine Veränderung des **RecView**-Objektes nicht zulässig. Eigenschaften und Befehle, die das **RecView**, die **Columns** oder **SubColumns** verändern würden, lösen den Laufzeitfehler **ErrHdlInvalid** aus. Das Abfragen von Eigenschaften sowie der Aufruf von **WinInfo()** sind zulässig.

Hat ein Objekt weder die Eigenschaft **HelpTip** gesetzt noch das Ereignis EvtHelpTip eingetragen, wird das Elternobjekt auf vorhandensein geprüft.

Bei deaktiven Objekten (**Disabled = true** oder **LstStyle = WinLstHeaderStatic**) wird kein Ereignis ausgelöst und kein HelpTip angezeigt.

### Beispiele

```
sub EvtHelpTip( aEvt : event; // Ereignis aButton : int;
```

**EvtHyphenate**

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aWord</u>	<u>alpha</u>	zu trennendes Wort
<u>aPrimLang</u>	<u>word</u>	Primäre Sprache
<u>aSubLang</u>	<u>word</u>	Sekundäre Sprache
<u>aExceed</u>	<u>int</u>	Index des Zeichens, das zum Umbruch führt
<u>aHyphResult</u>	<u>byte</u>	Resultat der Silbentrennung
<u>aHyphIndex</u>	<u>int</u>	Index des ersten Zeichens nach der Trennung
<u>aHyphChar</u>	<u>int</u>	Ersetzungszeichen
<u>Resultat</u>	<u>logic</u>	Silbentrennung durchführen? <u>Liste, Objekte.</u>

Siehe

**Ereignisbefehle**

 Das Ereignis steht nur unter Windows-Betriebssystemen zur Verfügung, die die Datei msftedit.dll mit mindestens der Version 5.41.15.1515 verwenden. Die Version kann im Info-Dialog des Designers ermittelt werden.

Das Ereignis wird ausgelöst, wenn eine Silbentrennung innerhalb eines RtfEdit-Objekts erfolgen soll.

Damit das Ereignis ausgelöst werden kann, muss es beim Application-Objekt mit der Anweisung WinEvtProcNameSet() eingetragen werden. Beim RtfEdit-Objekt muss die Eigenschaft HyphMarginTwips gesetzt sein.

**Definition des Funktionskopfs:**

```
sub EvtHyphenate(  aEvt           : event;          // Ereignis, aEvt:Obj = _App  aWord
aWord
```

**aEvt**

In diesem Übergabeparameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben. In diesem Fall ist es das Application-Objekt.

**aWord**

Hier wird das Wort übergeben, das getrennt werden soll. Das Ereignis wird zu dem Zeitpunkt ausgelöst, in dem es zum Zeilenumbruch kommt. D. h. in aWord steht nicht das vollständige Wort, sondern nur der Teil des Wortes, der bisher eingegeben wurde. Die maximale Länge des Wortes muss angegeben werden. Wird hier nichts angegeben, können höchstens 80 Zeichen übergeben werden.

**aPrimLang und aSubLang**

## Kontakt

Hier wird die primäre und die sekundäre Sprache übergeben. Die übergebenen Werte können mit den \_LclLang... bzw. \_LclSubLang...-Konstanten verglichen werden.

### aExceed

Dies ist die Position des Zeichens, das zu dem Zeilenumbruch geführt hat. Der Trennstrich würde als vor diesem Zeichen stehen.

### aHyphResult, aHyphIndex und aHyphChar

Diese Parameter werden nicht vom Ereignis gesetzt, sondern müssen in der Ereignis-Funktion gesetzt werden. Sie bestimmen das Verhalten bei der Silbentrennung.

- aHyphResult # WinHyphNIL

Wird der Wert WinHyphNIL gesetzt, findet keine Silbentrennung statt. Weitere Angaben in den Parametern aHyphIndex und aHyphChar werden ignoriert.

- aHyphResult # WinHyphNormal

Wird der Wert WinHyphNormal gesetzt, findet eine normale Trennung der Silben statt. In dem Parameter aHyphIndex wird die Position angegeben, nach der getrennt werden soll. Der Wert von aHyphIndex liegt im Bereich 1 bis aExceed - 1.

- aHyphResult # WinHyphAddBefore

Wird der Wert WinHyphAddBefore gesetzt, findet eine normale Trennung der Silben statt. In dem Parameter aHyphIndex wird die Position angegeben, nach der getrennt werden soll. Der Wert von aHyphIndex liegt im Bereich 1 bis aExceed - 1. Vor dem Trennstrich wird der in aHyphChar angegebene Buchstabe eingefügt.

- aHyphResult # WinHyphChangeBefore

Wird der Wert WinHyphChangeBefore gesetzt, findet eine normale Trennung der Silben statt. In dem Parameter aHyphIndex wird die Position angegeben, nach der getrennt werden soll. Der Wert von aHyphIndex liegt im Bereich 1 bis aExceed - 1. Der Buchstabe vor dem Trennstrich wird durch den in aHyphChar angegebenen Buchstaben ersetzt.

- aHyphResult # WinHyphDeleteBefore

Wird der Wert WinHyphDeleteBefore gesetzt, findet eine normale Trennung der Silben statt. In dem Parameter aHyphIndex wird die Position angegeben, nach der getrennt werden soll. Der Wert von aHyphIndex liegt im Bereich 1 bis aExceed - 1. Der Buchstabe vor dem Trennstrich wird entfernt.

- aHyphResult # WinHyphChangeAfter

Wird der Wert WinHyphChangeAfter gesetzt, findet eine normale Trennung der Silben statt. In dem Parameter aHyphIndex wird die Position angegeben, nach der getrennt werden soll. Der Wert von aHyphIndex liegt im Bereich 1 bis aExceed - 1. Der Buchstabe hinter dem Trennstrich wird durch den in aHyphChar angegebenen Buchstaben ersetzt.

## Kontakt

- aHyphResult # WinHyphDelAndChange

Wird der Wert WinHyphDelAndChange gesetzt, findet eine normale Trennung der Silben statt. In dem Parameter aHyphIndex wird die Position angegeben, nach der getrennt werden soll. Der Wert von aHyphIndex liegt im Bereich 1 bis aExceed - 1. Die beiden Buchstaben vor dem Trennstrich werden durch den in aHyphChar angegebenen Buchstaben ersetzt.

### Resultat

Damit die durch aHyphResult, aHyphIndex und aHyphChar definierten Rückgabewerte wirksam werden, muss das Ereignis true zurückgeben. Ein Wert von false entspricht aHyphResult = WinHyphNIL.

Beispiele:

aWord	aHyphRes	aHyphIndex	aHyphChar	Trennung
'Zuckerstange'	<u>WinHyphNIL</u>	-	-	'Zuckerstange'
'Zuckerstange'	<u>WinHyphNormal</u>	6	-	'Zucker- stange'
'Zuckerstange'	<u>WinHyphChangeBefore 3</u>		StrToChar('k','Zuk- 1)	'kerstange' // (alte Rechtschreibung)
'Zuckerstange'	<u>WinHyphNormal</u>	2	-	'Zu- ckerstange' // (neue Rechtschreibung)

## Kontakt

### EvtLstDataInit

#### Ereignis vor Datensatzanzeige

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aID</u>	<u>int / bigint</u>	ID des Datensatzes oder Zeilennummer
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle, Beispiel,</u> <u>Ereignisabläufe RecList</u>

Das Ereignis wird ausgelöst, wenn ein Datensatz in einem Listen-Objekt gelesen wurde. Die Ausgabe des Datensatzes erfolgte noch nicht. In diesem Ereignis kann die Veränderung der Feldinhalte, das Lesen von verknüpften Dateiinhalten, das Anpassen der Spaltenfarbe in der aktuellen Zeile usw. erfolgen.

Definition des Funktionskopfes:

```
sub EvtLstDataInit( aEvt : event; // Ereignis aID : bigint;
```

Alternativer Funktionskopf:

```
sub EvtLstDataInit( aEvt : event; // Ereignis aID : int;
```

aEvt

In diesem Parameter wird der Deskriptor des aufrufenden Objekts übergeben. Dabei handelt es sich um das Objekt, bei dem die Ereignisfunktion eingetragen wurde.

aID

Der Parameter enthält entweder die Datensatz-ID des gerade gelesenen Datensatzes (bei den Objekten RecList und RecListPopup) oder die Zeilennummer (bei den Objekten DataList und DataListPopup).



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Änderungen der folgenden Eigenschaften beziehen sich in diesem Ereignis auf die aktuelle Zelle statt auf die gesamte Spalte:

- ClmColBkg
- ClmColFg
- ClmColFocusBkg
- ClmColFocusFg
- ClmColFocusOffBkg

## Kontakt

- ClmColFocusOffFg
- ClmTypeImage
- FmtBoolStyle
- FontAttr
- FontParent



Im Modern Theme Style wirken sich Änderungen der Farbeigenschaften, sowie der Eigenschaft FontAttr im EvtLstDataInit aus, sofern sich der gesetzte Wert von dem gespeicherten Wert in der Spalte unterscheidet. Wird der Wert in EvtLstDataInit auf den gespeicherten Wert gesetzt oder nicht verändert, wird die jeweilige Eigenschaft von dem Theme-Objekt verwendet. Die Eigenschaft FontAttr wirkt sich zusätzlich nur dann aus, wenn die Eigenschaft FontParent nicht gesetzt ist.

### Beispiel:

```
sub EvtLstDataInit(  aEvt      : event;      // Ereignis  aID      : bigint;    // Datensatz-ID oder Zeil
```

## Kontakt

### EvtLstEditActivate

Aktivierung des Eingabeobjektes beim Ändern einer Gruppe im RecView

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aGroup	handle	Deskriptor der Gruppe
aRecBuf	handle	Deskriptor des Datensatzpuffers
aEdit	handle	Eingabeobjekt
Resultat logic		Wird nicht ausgewertet
		Liste, Objekte, EvtLstEditStartGroup,
		EvtLstEditStartItem, Ereignisbefehle,
Siehe		Beispiel, Ereignisabläufe RecView,
		CodeLibrary - RecViewEdit

Das Ereignis wird ausgelöst, wenn im Ereignis EvtLstEditStartItem im Parameter

aResult WinRvwGroupEditContinue zurückgegeben wurde. Hier kann das Edit-Objekt angepasst werden.

Definition des Funktionskopfes:

```
sub EvtLstEditActivate( aEvt : event; // Ereignis aGroup : han
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aGroup

In diesem Parameter wird der Deskriptor der editierten Gruppe übergeben. Die Eigenschaften SelectorItem und SelectorSubItem sind entsprechend gesetzt, so dass das zu editierende Item identifiziert werden kann.

aRecBuf

Hier wird der Datensatzpuffer mit den aktuellen Feldinhalten des Datensatzes übergeben.

aEdit

Hier wird das zum Editieren bereitgestellte Eingabeobjekt übergeben. Je nach Datentyp des zu editierenden Inhaltes wird ein anderes Objekt bereitgestellt. Handelt es sich um Rtf-Text, wird ein RtfEdit-Objekt übergeben. In diesem Fall hat das Eingabeobjekt noch keinen Inhalt. Dieser muss beispielsweise mit WinRtfLoad() manuell geladen werden.

Im Fall eines Rtf-Textes kann auch eine bestehende Rtf-Toolbar mit dem Eingabeobjekt verknüpft werden. Dazu muss dem Eingabeobjekt ein eindeutiger Name gegeben werden und eine Verknüpfung zwischen der Toolbar und dem Objekt per ObjLink hergestellt werden.

## Kontakt

Nach dem Aufruf dieses Ereignisses wird das Eingabeobjekt sichtbar gemacht.

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Besonderheiten



Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

Die Eigenschaft StyleBorder, von dem in aEdit übergebenen Objekt, kann hier auf WinBorFrame gesetzt werden, um einen Frame-Rahmen anzuzeigen, wodurch das Eingabeobjekt vergrößert werden kann.

### Beispiel:

```
sub EvtLstEditActivate( aEvt : event;      // Ereignis    aGroup : han
```

## Kontakt

### EvtLstEditCommit

Edit-Objekt einer Liste verlassen

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aColumn</u>	<u>handle</u>	Deskriptor der Spalte
<u>aKey</u>	<u>int</u>	Verwendete Taste
<u>aFocusObject</u>	<u>handle</u>	Deskriptor des Fokus-Objekts
<u>Resultat</u>	<u>logic</u>	Wert in Liste übernehmen? <u>Liste, Objekte,</u> <u>EvtLstEditStart,</u> <u>EvtLstEditFinished,</u> <u>Ereignisbefehle Beispiel,</u> <u>Ereignisabläufe WinLstEdit()</u>
Siehe		

Dieses Ereignis wird aufgerufen, wenn das Edit-Feld in einer Liste verlassen werden soll. Hier kann entschieden werden, ob der eingetragene Wert in die Liste übernommen werden soll.



Ein Eintrag, der in die Liste übernommen wird, ist nicht automatisch in der Datenbank gespeichert. Die Speicherung von Datensätzen muss in dem Ereignis EvtLstEditFinished erfolgen.

Definition des Funktionskopfes:

```
sub EvtLstEditCommit( aEvt : event; // Ereignis aColumn : handle
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

Ausgehend von diesem Deskriptor kann mit der Anweisung WinInfo(..., WinLstEditObject) das Eingabe-Objekt ermittelt werden.

aColumn

In diesem Parameter wird der Deskriptor der Spalte übergeben, in der die Liste editiert wurde.

aKey

Hier steht der Code der Taste, mit der das Edit-Objekt verlassen wurde. Der Wert kann mit den symbolischen Konstanten für Tasten (WinKey...) verglichen werden.

Wurde das Feld durch einen Klick mit der Maus verlassen, wird hier 0 übergeben.

aFocusObject

Wird das Edit-Objekt mit der Maus verlassen, wird hier der Deskriptor des Objekts, das als nächstes den Fokus erhält, übergeben. Befindet sich dieses Objekt nicht in der CONZEPT 16-Applikation, wird 0 übergeben.

## Kontakt

### Resultat

Der Rückgabewert der Funktion entscheidet, ob der eingetragene Wert in die Liste übernommen wird (true) oder nicht (false). Wurde beim Befehl WinLstEdit() die Option WinLstEditClearChanged angegeben, wird bei einem Rückgabewert von true ebenfalls das Changed-Flag gesetzt. Das Changed-Flag wird bei dem Ereignis EvtLstEditFinished übergeben.

### Beispiel:

```
sub EvtLstEditCommit( aEvt          : event; // Ereignis  aColumn        : handle; // Spalte   aKey
```

## Kontakt

### EvtLstEditEndGroup

Abschließen der Bearbeitung einer Gruppe im RecView

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aGroup	handle	Deskriptor der Gruppe

aRecBuf handle Deskriptor des Datensatzpuffers

aAbort logic Abbruch-Anforderung

aResult int Eingabe-Resultat

Resultat logic Wird nicht ausgewertet

Liste, Objekte, EvtLstEditStartGroup,

EvtLstEditStartItem,

Siehe EvtLstEditEndItem, Ereignisbefehle,

Beispiel, Ereignisabläufe RecView

Das Ereignis wird nach dem Ereignis EvtLstEditEndItem ausgelöst. Weiterhin wird es auch aufgerufen, wenn im Ereignis EvtLstEditStartGroup der Parameter aResult auf WinRvwGroupEditAbort gesetzt wurde. Hier kann der Datensatz gespeichert oder entsperrt werden.

Definition des Funktionskopfes:

```
sub EvtLstEditEndGroup( aEvt : event; // Ereignis aGroup : han
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aGroup

In diesem Parameter wird der Deskriptor der editierten Gruppe übergeben. Die Eigenschaften SelectorItem und SelectorSubItem sind entsprechend gesetzt, so dass das zu editierende Item identifiziert werden kann.

aRecBuf

Hier wird der Datensatzpuffer mit den aktuellen Feldinhalten des Datensatzes übergeben.

aAbort

Dieser Parameter gibt an, ob der Benutzer die Bearbeitung abbricht (true) oder nicht (false). Sofern Datenbankfelder im Ereignis EvtLstEditEndItem verändert wurden, sollten diese hier in die Datenbank übertragen werden und eine eventuell im Ereignis EvtLstEditStartGroup vorgenommene Datensatzsperrre ebenfalls aufgehoben werden.



Wird ein Item verändert und mit oder + das nächste oder vorherige Item angesprungen, wird in jedem Fall false übergeben, auch wenn im Ereignis EvtLstEditEndItem der Parameter aResult auf WinRvwGroupEditAbort gesetzt wurde.

## Kontakt

### aResult

Dieser Parameter ist reserviert für zukünftige Erweiterungen und wird momentan nicht ausgewertet.

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Beispiel:

```
sub EvtLstEditEndGroup( aEvt : event;      // Ereignis    aGroup : han
```

## Kontakt

### EvtLstEditEndItem

Editiervorgang eines Items oder SubItems im RecView beenden  
oder abbrechen

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aGroup	handle	Deskriptor der Gruppe
aRecBuf	handle	Deskriptor des <u>Datensatzpuffers</u>
aEdit	handle	Eingabe-Objekt
aAbort	logic	Abbruch-Anforderung
aResult	int	Eingabe-Resultat
Resultat	logic	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte, EvtLstEditStartItem,</u> <u>EvtLstEditEndGroup, Ereignisbefehle,</u> <u>Beispiel, Ereignisabläufe RecView</u>

Das Ereignis wird aufgerufen, wenn der Editiervorgang für das aktuelle Item oder SubItem beendet oder abgebrochen wird. In diesem Ereignis kann man den geänderten Wert in den Datensatzpuffer übertragen.

Definition des Funktionskopfes:

```
sub EvtLstEditEndItem( aEvt : event; // Ereignis aGroup : hand
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aGroup

In diesem Parameter wird der Deskriptor der editierten Gruppe übergeben. Die Eigenschaften SelectorItem und SelectorSubItem sind entsprechend gesetzt, so dass das zu editierende Item identifiziert werden kann.

aRecBuf

Hier wird der Datensatzpuffer mit den aktuellen Feldinhalten des Datensatzes übergeben.

aEdit

In diesem Parameter wird das Eingabeobjekt übergeben, dass zum Editieren bereit gestellt wurde. Der Inhalt des Eingabeobjektes muss manuell in die Datensatzpuffer übernommen werden.

aAbort

Dieser Parameter ist auf true gesetzt, wenn der Editiervorgang durch Verlassen des Eingabeobjektes oder durch Druck der -Taste abgebrochen wurde. Wird die Bearbeitung mit der in EditKeyEnd definierten Taste oder  beendet, wird false

## Kontakt

übergeben. In diesem Fall möchte der Anwender eine Änderung durchführen. Durch die Verwendung der -Taste kann in das nächste zu bearbeitende Item gesprungen werden, mit + in das vorherige.

### aResult

Über diesen Parameter kann ein Wert zurückgegeben werden, der definiert, ob im nachfolgenden Ereignis EvtLstEditEndGroup der Parameter aAbort = false (WinRvwGroupEditContinue) oder true (WinRvwGroupEditAbort) übergeben wird. Standardmäßig ist aResult mit Äquivalent von dem Parameter aAbort aus diesem Ereignis vorbelegt.

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Besonderheiten



Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

### Beispiel:

```
sub EvtLstEditEndItem( aEvt : event; // Ereignis aGroup : hand
```

Im ersten Fall wird der Inhalt für das Datenbankfeld 'faNewsTitle' mit dem Inhalt des Eingabeobjektes aktualisiert. Im zweiten Fall ein Datumsfeld (fdNewsDate) aktualisiert und im dritten Fall der Inhalt des Rtf-Textes übernommen.

## Kontakt

### EvtLstEditFinished

Ändern eines Spalteneintrages abgeschlossen

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aColumn</u>	<u>handle</u>	Deskriptor der Spalte
<u>aKey</u>	<u>int</u>	Verwendete Taste
<u>aRecID</u>	<u>int / bigint</u>	ID des Datensatzes oder Nummer der Zeile
<u>aClnChanged logic</u>		Spalteninhalt wurde geändert
<u>aEditChanged logic</u>		Inhalt des Eingabeobjektes wurde geändert (optional)
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
		<u>Liste, Objekte, EvtLstEditStart,</u> <u>EvtLstEditCommit,</u> <u>WinEvtProcessSet()</u> ,
Siehe		<u>WinEvtProcessGet()</u> , <u>Ereignisbefehle, Beispiel,</u> <u>Ereignisabläufe WinLstEdit()</u>

Ereignis wird nach jedem Verlassen eines Edit-Objektes innerhalb einer Liste aufgerufen. Das Edit-Objekt ist bereits aus dem Speicher entfernt. In diesem Ereignis kann entschieden werden, ob der geänderte Datensatz gespeichert werden soll oder zum Beispiel eine andere Spalte editiert werden soll.

**Definition des Funktionskopfes:**

```
sub EvtLstEditFinished( aEvt : event; // Ereignis aColumn : han
```

**Alternativer Funktionskopf:**

```
sub EvtLstEditFinished( aEvt : event; // Ereignis aColumn : han
```

**aEvt**

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts übergeben. Dies ist der Deskriptor des Objektes, in dem die Ereignisfunktion eingetragen wurde.

**aColumn**

In diesem Parameter wird der Deskriptor der Spalte übergeben, in der die Liste editiert wurde.

**aKey**

Hier steht der Code der Taste, mit der das Edit-Objekt verlassen wurde. Der Wert kann mit den symbolischen Konstanten für Tasten (WinKey...) verglichen werden.

## Kontakt

Bei einer Auswahl aus einem DataListPopup-Objekt wird in diesem Parameter WinKeySelect übergeben.

Wurde das Feld durch einen Klick mit der Maus verlassen, wird hier 0 übergeben.

### aRecID

Hier wird die Datensatz-ID des editierten Datensatzes übergeben. Wird eine DataList editiert, wird die aktuelle Zeilennummer übergeben.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

### aClmChanged

Mit diesem Übergabeparameter kann entschieden werden, ob Felder verändert wurden. Das Changed-Flag wird durch den Befehl WinLstEdit() mit der Option WinLstEditClearChanged zurückgesetzt und gesetzt, sobald das Ereignis EvtLstEditCommit den Wert true zurück gibt.

### aEditChanged

In diesem Übergabeparameter wird übergeben, ob sich der Inhalt des Eingabeobjektes nach Erstellung bzw. nach Durchführung des Ereignisses EvtLstEditStart verändert hat.

## Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Beispiel:

```
sub EvtLstEditFinished( aEvt : event; // Ereignis aColumn : handle; // Spalte aK
```

## Kontakt

### EvtLstEditStart

Spalteneintrag einer Liste ändern

	Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>		Ereignisinformationen
<u>aColumn</u>	<u>handle</u>	Deskriptor der Spalte	
<u>aEdit</u>	<u>handle</u>	Deskriptor des Eingabe-Objekts	
<u>aList</u>	<u>handle</u>	Deskriptor des <u>DataListPopup-Objekts</u>	
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet	
		<u>Liste, Objekte,</u>	
		<u>EvtLstEditCommit,</u>	
<u>Siehe</u>		<u>EvtLstEditFinished,</u>	
		<u>Ereignisbefehle, Beispiel,</u>	
		<u>Ereignisabläufe WinLstEdit()</u>	

Das Ereignis wird ausgelöst, wenn ein Eintrag in einer Liste editiert werden soll. Das Editieren wird durch den Befehl WinLstEdit() gestartet. Hier können Vorbereitungen zum Editieren des Datensatzes oder der Zeile in einer DataList getroffen werden.

Definition des Funktionskopfes:

```
sub EvtLstEditStart( aEvt : event; // Ereignis aColumn : handle
```

**aEvt**

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aColumn**

In diesem Parameter wird der Deskriptor der editierten Spalte übergeben.

**aEdit**

Hier wird der Deskriptor des erzeugten Edit-Objekts übergeben.

**aList**

Wurde der Befehl WinLstEdit() ohne die Parameter WinLstEditLst oder WinLstEditLstAlpha aufgerufen, wird 0 übergeben. Im anderen Fall steht hier der Deskriptor des erzeugten DataListPopup-Objekts.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Beispiel:**

```
sub EvtLstEditStart( aEvt : event; // Ereignis aColumn : handle; // Spalte aEdi
```

## Kontakt

Der Aufbau einer zweispaltigen **DataSet** (WinLstEditLstAlpha) wird wie folgt vorgenommen:

```
...// erste SpalteList->WinLstDatLineAdd('Herr');// zweite SpalteList->WinLstCellSet('Mister',
```

## Kontakt

### EvtLstEditStartGroup

#### Gruppe eines RecViews ändern

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen Deskriptor der

#### aGroup handle Gruppe

<u>aRecBuf</u>	<u>handle</u>	Deskriptor des <u>Datensatzpuffers</u>
<u>aResult</u>	<u>int</u>	Eingabe-Resultat
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
		<u>Liste, Objekte,</u>
		<u>EvtLstEditStartItem,</u>
<u>Siehe</u>		<u>EvtLstEditEndGroup,</u>
		<u>Ereignisbefehle, Beispiel,</u>
		<u>Ereignisabläufe RecView</u>

Das Ereignis wird ausgelöst, wenn ein Eintrag in einem RecView-Objekt editiert werden soll. Das Editieren wird durch die in EditKeyStart und EditMouseStart definierten Tastatur- und Mausaktionen ausgelöst. Hier können Vorbereitungen zum Editieren des Datensatzes getroffen oder das Editieren für alle oder bestimmte Items der Gruppe unterbunden werden.

#### Definition des Funktionskopfes:

```
sub EvtLstEditStartGroup( aEvt : event; // Ereignis aGroup : h
```

#### aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

#### aGroup

In diesem Parameter wird der Deskriptor der editierten Gruppe übergeben. Durch Setzen der Eigenschaften SelectorItem und SelectorSubItem können die Eigenschaften eines beliebigen Items bzw. SubItems der Gruppe gelesen werden.

#### aRecBuf

Hier wird der Datensatzpuffer mit den aktuellen Feldinhalten des Datensatzes übergeben.

#### aResult

Über diesen Parameter kann ein Wert zurückgegeben werden, der definiert, ob der Editievorgang gestartet (WinRvwGroupEditContinue) oder abgebrochen werden soll (WinRvwGroupEditAbort). Standardmäßig ist aResult mit WinRvwGroupEditContinue vorbelegt. Die Rückgabe von dieser Konstanten führt zum Auslösen des Ereignisses EvtLstEditStartItem. Wird WinRvwGroupEditAbort

## Kontakt

zurückgegeben, wird anschließend das Ereignis EvtLstEditEndGroup aufgerufen.

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Besonderheiten

-  Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

Beispiel:

```
sub EvtLstEditStartGroup( aEvt : event;      // Ereignis aGroup : h
```

## Kontakt

### EvtLstEditStartItem

Item oder SubItem eines RecViews ändern

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aGroup</u>	<u>handle</u>	Deskriptor der Gruppe
<u>aRecBuf</u>	<u>handle</u>	Deskriptor des Datensatzpuffers
<u>aResult</u>	<u>alpha</u>	Eingabe-Resultat
<u>aOptions int</u>		Eingabe-Optionen (optional)
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
		<u>Liste, Objekte,</u> <u>EvtLstEditStartGroup,</u> <u>EvtLstEditActivate,</u>
<u>Siehe</u>		<u>EvtLstEditEndItem,</u> <u>Ereignisbefehle, Beispiel,</u> <u>Ereignisabläufe RecView,</u>
		CodeLibrary - RecViewEdit

Das Ereignis wird ausgelöst, wenn im Ereignis EvtLstEditStartGroup im Parameter aResult WinRvwGroupEditContinue zurückgegeben wurde. Es leitet den Editiervorgang für das in aGroup übergebene Item oder SubItem ein. Hier kann das Editieren für bestimmte Items der Gruppe unterbunden werden.

Definition des Funktionskopfes:

```
sub EvtLstEditStartItem( aEvt : event; // Ereignis aGroup : ha
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aGroup

In diesem Parameter wird der Deskriptor der editierten Gruppe übergeben. Die Eigenschaften SelectorItem und SelectorSubItem sind entsprechend gesetzt, so dass das zu editierende Item identifiziert werden kann.

aRecBuf

Hier wird der Datensatzpuffer mit den aktuellen Feldinhalten des Datensatzes übergeben.

aResult

Über diesen Parameter kann ein Wert zurückgegeben werden, der definiert, ob der Editiervorgang gestartet (WinRvwGroupEditContinue) oder abgebrochen werden soll (WinRvwGroupEditAbort). Mit der Konstanten WinRvwGroupEditSkipItem kann der Editiervorgang für ein Item übersprungen werden. In diesem Fall wird das Ereignis

## Kontakt

für das nächste Item in der Reihenfolge von VisibleOrder ausgeführt. Standardmäßig ist aResult mit WinRvwGroupEditContinue vorbelegt. Die Rückgabe von dieser Konstanten führt zum Auslösen des Ereignisses EvtLstEditActivate. Wird WinRvwGroupEditAbort zurückgegeben, wird anschließend das Ereignis EvtLstEditEndItem aufgerufen.

### aOptions

Dieser Parameter entscheidet darüber, ob für das durch aGroup spezifizierte GroupItem eine Popup-Liste erstellt werden soll oder nicht.

Dem Argument kann einer der folgenden Werte zugewiesen werden:

- \_WinRvwEditNone

Es wird keine Populiste erstellt. Dies ist das Standardverhalten.

- \_WinRvwEditLst

Es wird eine Populiste mit einer Spalte, die dem Datentyp des GroupItem-Objektes entspricht erstellt. Beim Aufklappen der Populiste wird der Eintrag selektiert, der dem Inhalt des Eingabefeldes entspricht. Der selektierte Eintrag wird beim Zuklappen ins Eingabefeld übertragen.

- \_WinRvwEditLstAlpha

Es wird eine Populiste mit zwei Spalten erstellt. Die erste Spalte besitzt denselben Datentyp, wie das GroupItem-Objekt, ist jedoch unsichtbar. Die zweite Spalte ist vom Typ alpha. Die Einträge dieser Spalte werden angezeigt. Ein Eintrag wird selektiert, wenn der Inhalt des Eingabefeldes dem Inhalt der 1. Spalte entspricht. Der Wert der 1. Spalte wird beim Zuklappen in das Eingabefeld übertragen.

Wird eine der Optionen WinRvwEditLst oder WinRvwEditLstAlpha gesetzt, dann ist die Populiste zunächst leer. Diese kann im Ereignis EvtLstEditActivate gefüllt werden. Dort wird der Deskriptor des Eingabefeldes übergeben. Ausgehend hiervon kann die Populiste ermittelt werden.

```
sub GetPopupFromEdit( aEdit : handle; ) : handle; { return (aEdit->WinInfo(_WinFir
```

Anschließend kann über die bekannten Befehle die Populiste gefüllt werden:

```
tPopupList # aEdit->GetPopupListFromEdit(); tPopupList->WinLstDatLineAdd('S'); tPopupList->Wi
```

Für den Datentyp logic wird eine automatisch gefüllte Liste generiert, die in Abhängigkeit von aOptions unterschiedlich ausfällt:

- \_WinRvwEditLst

Es wird eine einspaltige Liste generiert. Sie enthält zwei Einträge, die mit Häkchen dargestellt werden.

- \_WinRvwEditLstAlpha

## Kontakt

Es wird eine zweispaltige Liste generiert. Die erste Spalte ist unsichtbar. Die zweite Spalte enthält zwei Einträge, die in Abhängigkeit von der Eigenschaft FmtBoolString gesetzt werden.

Der in der Liste selektierte Wert für true bzw. false kann mit der folgenden Funktion ausgehend vom Eingabefeld ermittelt werden.

```
sub GetSelectedLogicValue( aEdit : handle; ) : logic; local { tList
```

## Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

## Besonderheiten



Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

Beispiel:

```
sub EvtLstEditStartItem( aEvt : event; // Ereignis aGroup : ha
```

Das SubItem im Beispiel stellt ein Bild dar und gibt über das Resultat WinRvwGroupEditAbort zurück, dass dieses nicht editiert werden kann.

## Kontakt

### EvtLstGroupArrange

Ereignis nach der Initialisierung einer Gruppe \_\_\_\_\_

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen

aRecBuf handle Datensatzpuffer

aGroup handle Gruppe

aViewId int Nummer des Views \_\_\_\_\_

Resultat logic Wird nicht ausgewertet

Liste, Objekte,

Siehe EvtLstGroupInit,

Ereignisbefehle

Das Ereignis wird aufgerufen, nachdem eine Gruppe durch EvtLstGroupInit

initialisiert wurde. Die Ausgabe der Gruppe erfolgte noch nicht. In diesem Ereignis können SubItems neu positioniert werden.

Definition des Funktionskopfes:

```
sub EvtLstGroupArrange( aEvt : event; // Ereignis aRecBuf : han
```

aEvt

In diesem Parameter wird der Deskriptor des aufrufenden Objekts übergeben. Dabei handelt es sich um das Objekt, bei dem die Ereignisfunktion eingetragen wurde.

aRecBuf

In aRecBuf wird der Deskriptor des Datensatzpuffers angegeben, über den das View des RecViews die Datensätze liest.

aGroup

Der Parameter enthält den Deskriptor der Gruppe, die angezeigt werden soll.

In diesem Ereignis hat die Gruppe die Eigenschaften ContentWidth, ContentHeight, ContentLeft und ContentTop. Die Eigenschaft ContentHeight kann beim Item und SubItem geändert werden. Sie enthält die Höhe des Inhaltes von dem Item oder SubItem. Bei SubItems können zusätzlich die Eigenschaften ContentLeft, ContentTop und ContentWidth verändert werden. Bei Items sind die drei Eigenschaften nur lesbar. Diese Eigenschaften definieren die Position und die Breite des Items oder SubItems relativ zur Gruppe.

aViewId

Dieser Parameter enthält die Nummer des Views, das die Gruppe zur Anzeige initialisiert hat. Ein RecView kann über die Eigenschaft SplitStyle in bis zu vier Views geteilt werden. Dadurch kann eine unterschiedliche Darstellung der Items und SubItems pro View erreicht werden.

Resultat

## Kontakt

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Besonderheiten

Um ein Item oder SubItem auszuwählen, müssen die Eigenschaften SelectorItem und SelectorSubItem gesetzt werden. Anschließend können die Eigenschaften des Items beziehungsweise SubItems geändert werden.



Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Alle anderen Eigenschaften, außer SelectorItem, SelectorSubItem, ContentWidth, ContentHeight, ContentLeft und ContentTop sowie Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

Auf die Eigenschaften AreaLeft und AreaTop kann in diesem Ereignis nicht zugegriffen werden.

Beispiel:

```
sub EvtLstGroupArrange( aEvt : event;      // Ereignis    aRecBuf : han
```

## Kontakt

### EvtLstGroupInit

Ereignis vor Gruppenanzeige

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aRecBuf	handle	Datensatzpuffer
aGroup	handle	Gruppe
aViewId	int	Nummer des Views

Resultat logic      Wird nicht ausgewertet

Siehe      Liste, Objekte,  
Ereignisbefehle,  
Ereignisabläufe RecView

Das Ereignis wird aufgerufen, wenn eine Gruppe zum ersten Mal angezeigt werden soll (z. B. weil der Anwender diese in den sichtbaren Bereich scrollt), oder sie nicht mehr im Cache des RecView vorhanden ist. Die Ausgabe der Gruppe erfolgte noch nicht. In diesem Ereignis kann die Veränderung der Feldinhalte, das Lesen von verknüpften Dateiinhalten, das Anpassen der Itemfarbe in der aktuellen Gruppe usw. erfolgen.

Definition des Funktionskopfes:

```
sub EvtLstGroupInit( aEvt : event; // Ereignis aRecBuf : handle
```

aEvt

In diesem Parameter wird der Deskriptor des aufrufenden Objekts übergeben. Dabei handelt es sich um das Objekt, bei dem die Ereignisfunktion eingetragen wurde.

aRecBuf

In aRecBuf wird eine Kopie des internen Datensatzpuffers angegeben, über den das View des RecViews die Datensätze liest. Änderungen am Inhalt des übergebenen Datensatzpuffers wirken sich nicht auf die dargestellten Daten aus.

aGroup

Der Parameter enthält den Deskriptor der Gruppe, die angezeigt werden soll. Durch Setzen der Eigenschaft ExpireTime der Gruppe kann definiert werden, wann das Ereignis EvtLstGroupInit frühestens für diese Gruppe wieder aufgerufen wird.

Weiterhin kann die Vorder- und die Hintergrundfarbe für die Gruppe mit den Eigenschaften GroupColFg und GroupColBkg gesetzt werden. Bei allen Items, bei denen die Eigenschaft ClmColFg bzw. ClmColBkg auf WinColParent gesetzt ist, wird die jeweilige Farbe der Gruppe übernommen.

aViewId

Dieser Parameter enthält die Nummer des Views, das die Gruppe zur Anzeige initialisiert. Ein RecView kann über die Eigenschaft SplitStyle in bis zu vier Views geteilt werden. Dadurch kann eine unterschiedliche Darstellung der Items und

## Kontakt

SubItems pro View erreicht werden.

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Besonderheiten

Um ein Item oder SubItem auszuwählen, müssen die Eigenschaften SelectorItem und SelectorSubItem gesetzt werden. Anschließend können die Eigenschaften des Items beziehungsweise SubItems geändert werden.



Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.

Die Eigenschaft Caption... kann bei den Items mit dem jeweiligen Typ direkt gesetzt werden. Somit werden keine Hilfsfelder mehr benötigt. Die Eigenschaft ContentSource muss dazu auf WinContentSourceField gesetzt sein.

### Beispiel:

```
sub EvtLstGroupInit( aEvt : event; // Ereignis aRecBuf : handle
```

## Kontakt

### EvtLstRecControl

Überspringen einzelner Datensätze

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aRecID</u>	<u>int</u> <small>bigint</small>	/ ID des Datensatzes
<u>Resultat</u>	<u>logic</u>	Datensatz anzeigen?
		<u>Liste</u> , <u>Objekte</u> , <u>DbFilter</u> , <u>DbSelection</u> ,
Siehe		<u>Ereignisbefehle</u> , <u>Ereignisabläufe RecList</u> , <u>Ereignisabläufe RecView</u>

Das Ereignis wird ausgelöst, bevor ein Datensatz dargestellt wird. Es kann verwendet werden, um bestimmte Datensätze im RecList- oder RecListPopup-Objekt nicht darzustellen.



Aus Performancegründen sollte dieses Ereignis nur dann verwendet werden, wenn die Anzahl der Sätze, die angezeigt wird, überwiegt und keine allzu großen Lücken vorhanden sind, da jeder Satz der übersprungen wird, auch gelesen werden muss.

Definition des Funktionskopfes:

```
sub EvtLstRecControl( aEvt : event; // Ereignis aRecID : bigint
```

Alternativer Funktionskopf:

```
sub EvtLstRecControl( aEvt : event; // Ereignis aRecID : int;
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes übergeben. Dies ist der Deskriptor des Objektes, in dem die Ereignisfunktion eingetragen wurde.

aRecID

Dies ist die Datensatz-ID des Datensatzes, der angezeigt werden soll.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

Resultat

Der Rückgabewert bestimmt, ob der Datensatz angezeigt wird. Bei true wird der Datensatz angezeigt, bei false nicht.



Der Datensatzpuffer der Datei, die zum Aufbau des RecList-Objekts benötigt wird, darf nicht verändert werden. Bei Änderungen kann nicht mehr die korrekte Reihenfolge ermittelt werden und es kann dazu kommen, dass Datensätze übersprungen oder eine Endlosschleife programmiert wird. Muss der

## Kontakt

entsprechende Feldpuffer verändert werden, muss dafür Sorge getragen werden, dass der ursprüngliche Zustand des Feldpuffers wieder hergestellt wird, bevor die Funktion beendet wird (siehe RecBufCopy()).

Beispiele:

```
sub EvtLstRecControl( aEvt    : event;      // Ereignis  aRecID : bigint;     // Datensatz-ID): logic
```

## Kontakt

### EvtLstSelect

#### Änderung der selektierten Zeile

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aID</u>	<u>int / bigint</u>	ID des Datensatzes oder Nummer der Zeile
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet

Siehe

Liste, Objekte, Ereignisbefehle,

Ereignisabläufe RecList,  
Ereignisabläufe RecView

Das Ereignis wird ausgelöst, wenn sich die selektierte Zeile in einem Listen-Objekt geändert hat. Dies ist unabhängig davon, ob dies per Maus oder Tastatur geschehen ist. Das Ereignis wird auch beim Laden des Frames ausgelöst.

Im Zusammenhang mit einer ComboBox (RecListPopup, DataListPopup oder StoListPopup) wird das Ereignis erst mit dem Schließen der ComboBox ausgelöst.

Beim Auslösen des Ereignisses werden die Feldpuffer der Felder, die in den Spalten einer RecList enthalten sind, mit den Werten des selektierten Datensatzes gefüllt. Dieses Verhalten kann durch das Löschen des Flags WinLstRecSelectBuffer in der Eigenschaft LstFlags unterbunden werden. Der Datensatz kann dann über die Record-ID gelesen werden.



Das Ereignis wird ebenfalls ausgelöst, wenn kein Datensatz in der Liste dargestellt wird oder ein Update auf die Liste durchgeführt wird. Im Übergabeparameter aID wird dann der Wert 0 übergeben. Das Auslösen des Ereignisses EvtLstSelect bei einem Update auf die RecList kann durch die Option WinLstRecEvtSkip unterbunden werden.

#### Definition des Funktionskopfes:

```
sub EvtLstSelect( aEvt : event; // Ereignis aID : bigint;
```

#### Alternativer Funktionskopf:

```
sub EvtLstSelect( aEvt : event; // Ereignis aID : int;
```

#### aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objektes, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

#### aID

Beim Reclist- bzw. beim RecListPopup-Objekt wird in aID die Datensatz-ID des selektierten Datensatzes übergeben. Beim DataList- bzw. DataListPopup-Objekt wird die Nummer der selektierten Zeile übergeben. Beim StoList- bzw. StoListPopup-Objekt

## Kontakt

hat die Nummer keine Bedeutung.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

### Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.



Der Datensatzpuffer der Datei, die zum Aufbau des RecList-Objekts benötigt wird, darf nicht verändert werden. Bei Änderungen kann nicht mehr die korrekte Reihenfolge ermittelt werden und es kann dazu kommen, dass Datensätze übersprungen oder eine Endlosschleife programmiert wird. Muss der entsprechende Feldpuffer verändert werden, muss dafür Sorge getragen werden, dass der ursprüngliche Zustand des Feldpuffers wieder hergestellt wird, bevor die Funktion beendet wird (siehe RecBufCopy()).

### Beispiele:

```
// Fenster aktualisierensub EvtLstSelect( aEvt    : event;      // Ereignis  aID      : bigint;    //  
// Ereignisaufruf mit Sicherung der Feldpuffersub EvtLstSelect( aEvt    : event;      // Ereignis
```

## Kontakt

### EvtLstSelectRange

#### Selektion eines Bereichs

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aSelDataObj</u>	<u>handle</u>	<b>Deskriptor des SelectionData-Objekts</b>
<u>aID1</u>	<u>int / bigint</u>	Erster Datensatz der Selektion
<u>aID2</u>	<u>int / bigint</u>	Letzter Datensatz der Selektion
<u>aDirection</u>	<u>int</u>	Richtung der Selektion
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet <u>Liste, Objekte,</u>

Siehe

#### Ereignisbefehle

Dieses Ereignis wird ausgelöst, wenn in dem Objekt eine Mehrfachselektion mit gedrückter -Taste ausgeführt wird.

Definition des Funktionskopfes:

```
sub EvtLstSelectRange( aEvt : event; // Ereignis aSelDataObj : hand
```

Alternativer Funktionskopf:

```
sub EvtLstSelectRange( aEvt : event; // Ereignis aSelDataObj : hand
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aSelDataObj

In diesem Übergabeparameter wird das SelectionData-Objekt übergeben. Über das Objekt können alle bisher selektierten Datensätze ermittelt und die zu selektierenden Datensätze hinzugefügt werden.

aId1 und aId2

In diesen Parametern wird die Datensatz-ID des ersten und des letzten Datensatzes angegeben, die selektiert wurden. Der Programmierer kann durch Lesen der Datensätze und der Schlüsselinformation aus dem auslösenden RecList-Objekt alle dazwischenliegenden Datensätze ermitteln.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

aDirection

## Kontakt

Hier wird die Richtung, in der die Datensätze selektiert wurden übergeben. Der Parameter kann folgende Werte annehmen:

**WinMsdSelDown** Die Datensätze sind von oben nach unten in der Liste selektiert worden.

**WinMsdSelUp** Die Datensätze sind von unten nach oben in der Liste selektiert worden.

## Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Beispiel:

```
sub EvtLstSelectRange( aEvt : event; // Ereignis aSelDataObj : handle; // SelectD  
else { // Dateinummer ermitteln tFileNo # aEvt:Obj->wpDbFileNo; // Schlüsselnummer bz
```

 Es ist zu beachten, dass möglicherweise nicht alle Datensätze, die in der Liste selektiert sind, in die Liste der Datensätze übernommen werden. Die Datensätze, die zwischenzeitlich gelöscht wurden, werden nicht übernommen. Wird der letzte Datensatz der Selektion gelöscht, terminiert die Schleife nicht rechtzeitig. Ein Löschen kann durch das Sperren des Datensatzes verhindert werden.

## Kontakt

### EvtLstViewInit

Ereignis bei erstmaliger Initialisierung eines Views

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aRecBuf</u>	<u>handle</u>	Datensatzpuffer
<u>aViewId</u>	<u>int</u>	Nummer des Views
<u>aUpdateMode</u>	<u>int</u>	Initialisierungsmodus
		View mit dem übergebenen
<u>Resultat</u>	<u>logic</u>	Initialisierungsmodus
		aktualisieren?

Siehe

Liste, Objekte, Ereignisbefehle,

Ereignisabläufe RecView

Das Ereignis wird aufgerufen, wenn ein View erstellt und initialisiert wird. Dieses Ereignis bestimmt den Aufbau der Liste und den selektierten Satz.

Definition des Funktionskopfes:

```
sub EvtLstViewInit( aEvt : event; // Ereignis aRecBuf : handle;
```

**aEvt**

In diesem Parameter wird der Deskriptor des aufrufenden Objekts übergeben. Dabei handelt es sich um das Objekt, bei dem die Ereignisfunktion eingetragen wurde.

**aRecBuf**

In aRecBuf wird der Deskriptor des Datensatzpuffers angegeben, über den das View des RecViews die Datensätze liest.

**aViewId**

Dieser Parameter enthält die Nummer des Views, das initialisiert wird. Ein RecView kann über die Eigenschaft SplitStyle in bis zu vier Views geteilt werden. Dadurch kann eine unterschiedliche Darstellung der Items und SubItems pro View erreicht werden.

**aUpdateMode**

In diesem Parameter wird der gewünschte Initialisierungsmodus zurückgegeben. Der Parameter kann auf folgende Werte gesetzt werden:

- WinRvwUpdateFromFirst - Die erste angezeigte Zeile ist der erste vorhandene Datensatz.
- WinRvwUpdateFromLast - Die letzte angezeigte Zeile ist der letzte vorhandene Datensatz.
- WinRvwUpdateFromTop - Der erste angezeigte Eintrag bleibt unverändert.
- WinRvwUpdateFromRecBuf - Der erste angezeigte Eintrag wird durch den Inhalt von aRecBuf bestimmt.
- WinRvwUpdateDoSelect - Der positionierte Datensatz wird selektiert.

## Kontakt

Die Konstanten \_WinRvwUpdateFrom... können mit \_WinRvwUpdateDoSelect kombiniert werden, um den Datensatz zu selektieren. Wird diese Konstante nicht gesetzt, wird die Selektion aufgehoben.

-  Wird in **aUpdateMode 0** zurückgegeben, findet ein Aufbau des Views ab dem ersten Datensatz statt.

### Resultat

Der Rückgabewert der Funktion entscheidet darüber, ob das View mit dem angegebenen Initialisierungsmodus aktualisiert wird (true) oder das RecView ab dem ersten Datensatz aufgebaut wird.

### Besonderheiten

-  Während dieses Ereignisses ist eine Veränderung des RecView-Objektes nicht zulässig. Eigenschaften und Befehle, die das RecView, die Columns oder SubColumns verändern würden, lösen den Laufzeitfehler ErrHdlInvalid aus. Das Abfragen von Eigenschaften sowie der Aufruf von WinInfo() sind zulässig.
-  Das Ereignis wird nicht aufgerufen, wenn aufgrund nicht gesetzter Eigenschaften wie DbFileNo, DbLinkFileNo oder DbKeyNo keine Datensätze vorhanden sind.

### Beispiel:

```
sub EvtLstViewInit(  aEvt          : event;      // Ereignis  aRecBuf        : handle;
```

## Kontakt

### EvtNodeExpand

#### Schließen / Öffnen eines Knotens

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen

aNode handle Deskriptor des  
Knotens

aCollapse logic Schließen (true) oder  
Öffnen (false)

Resultat logic Aktion durchführen?

Siehe Liste, Objekte,  
Ereignisbefehle

Dieses Ereignis wird beim Schließen oder Öffnen eines Knotens ausgelöst.

Definition des Funktionskopfs:

```
sub EvtNodeExpand( aEvt : event; // Ereignis aNode : handle;
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts übergeben. Dies ist der Deskriptor des Objekts, in dem die Ereignisfunktion eingetragen wurde.

aNode

aNode enthält den Deskriptor des Knotens, der geöffnet oder geschlossen werden soll. Während in aEvt der Deskriptor des TreeView übergeben wird, steht in aNode der Deskriptor des Knotens.

aCollapse

Wird in aCollapse true übergeben soll der Knoten geschlossen, bei der Übergabe von false geöffnet werden.

Resultat

Der Rückgabewert entscheidet darüber, ob die Aktion Schließen oder Öffnen ausgeführt wird (true) oder nicht (false).

## Kontakt

### EvtNodeSearch

**TreeNode** wird bei **TreeView-Suche** geprüft

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aNode</u>	<u>handle</u>	Deskriptor des aktuell zu prüfenden <b>TreeNode</b> -Objektes
<u>aPattern alpha</u>	<u>Suchtext</u>	
<u>aFlags</u>	<u>int</u>	Suchoptionen
<u>aAction</u>	<u>int</u>	Aktion
<u>Resultat logic</u>		Suche fortsetzen? <u>Liste, Objekte, Ereignisbefehle,</u>

Siehe WinTreeNodeSearch()

Dieses Ereignis wird für jedes **TreeNode**-Objekt aufgerufen, das bei der Suche durchlaufen wird.

**Definition des Funktionskopfes:**

```
sub EvtNodeSearch( aEvt : event; // Ereignis aNode : handle;
```

**aEvt**

In aEvt vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts übergeben.

**aNode**

Im Argument aNode wird der Deskriptor des aktuellen **TreeNode**-Objektes übergeben, der verglichen werden soll.

**aPattern**

In diesem Argument wird der eingegebene Suchtext übergeben.

**aFlags**

Das Argument aFlags beinhaltet die Optionen, die an WinTreeNodeSearch() übergeben wurden, wenn die Suche über diese Funktion gestartet wurde. Beim Start über SearchKeyStart werden die SearchFlags übergeben. Zusätzlich kann aFlags WinTreeNodeSearchWrap enthalten, wenn der Anfang (WinTreeNodeSearchUp) oder das Ende des **TreeView**-Objektes bzw. des Eltern-Knotens (mit Option WinTreeNodeSearchChildrenOnly) erreicht wurde.

**aAction**

Im Argument aAction wird das weitere Verhalten der Suche definiert. Beim Aufruf des Ereignisses wird einer der Werte vorbelegt. In diesem Ereignis kann das Argument verändert werden um den weiteren Verlauf der Suche zu beeinflussen. Folgende Konstanten sind möglich:

## Kontakt

- WinTreeNodeSearchSkip

Der aktuelle TreeNode wird ignoriert. Dies ist der Standardwert, wenn der Suchtext nicht gefunden wird.

- WinTreeNodeSearchFound

Der aktuelle TreeNode gilt als Treffer. Die Suche wird beendet und der Knoten selektiert, sofern in aFlags nicht WinTreeNodeSearchNoSelect gesetzt ist.

- WinTreeNodeSearchSkipChildren

Die Kinder des in aNode übergebenen TreeNode werden nicht verglichen und die Suche läuft weiter.

Ist in aFlags WinTreeNodeSearchUp gesetzt, wird nur der aktuelle TreeNode ignoriert.

Wenn der Vergleich mit der Caption bzw. Custom, sofern in den Suchoptionen (aFlags) definiert, eine Übereinstimmung liefert, steht in aAction der Wert WinTreeNodeSearchFound und muss, damit der Knoten als Treffer gilt und selektiert wird nicht verändert werden. Somit gibt aAction Auskunft darüber ob der interne Vergleich ein Treffer war (WinTreeNodeSearchFound) oder nicht (WinTreeNodeSearchSkip).

### Resultat

Wird true zurückgegeben, wird die Suche fortgesetzt, bis in aAction der Wert WinTreeNodeSearchFound zurückgegeben oder das Startobjekt der Suche wieder erreicht wird. Bei false wird die Suche abgebrochen und der Wert in aAction ignoriert.



Das Ereignis EvtNodeSearch wird, für die Tasten SearchKeyNext und SearchKeyPrev, vor dem Ereignis EvtKeyItem ausgelöst.

Beispiel:

```
sub EvtNodeSearch( aEvt : event;      // Ereignis aNode : handle;
```

**EvtNodeSelect**

Knoten wurde selektiert

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aNode	handle	Selekterter Knoten
Resultat	logic	Wird nicht ausgewertet
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>

Dieses Ereignis wird ausgelöst, wenn ein Knoten des Objekts TreeView selektiert wird.

Definition des Funktionskopfs:

```
sub EvtNodeSelect( aEvt : event; // Ereignis aNode : handle;
```

**aEvt**

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aNode**

In diesem Parameter wird der Deskriptor des Knotens übergeben, der selektiert wurde. Während in aEvt der Deskriptor des TreeView-Objekts übergeben wird, steht in aNode der Deskriptor des Knotens.

**Resultat**

Der Rückgabewert wird bei diesem Ereignis nicht ausgewertet.



Mit WinInfo(0, WinNodeMouseSelect) kann die Maustaste ermittelt werden, mit der der Knoten im TreeView selektiert wurde.

## Kontakt

### EvtPageSelect

Notebook-Seite selektieren

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aPage</u>	<u>handle</u>	Deskriptor der Notizbuchseite
<u>aSelecting</u>	<u>logic</u>	Seite aufgerufen / verlassen
		Deskriptor der nächsten
<u>aPageNew</u>	<u>handle</u>	Notizbuchseite (optional)

<u>Resultat</u>	<u>logic</u>	Aktion durchführen?
Siehe	<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>	

Das Ereignis wird von dem Objekt Notebook ausgelöst, wenn eine Notebook-Seite aufgerufen oder die aktuelle Seite verlassen wird.

Definition des Funktionskopfs:

```
sub EvtPageSelect( aEvt : event;      // Ereignis   aPage : handle;
```

**aEvt**

Die Funktion hat einen Übergabeparameter vom Typ event. In aEvt wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aPage**

Der Deskriptor der aktivierten oder deaktivierten Notebook-Seite.

**aSelecting**

Gibt an, ob eine Seite aufgerufen oder verlassen wurde. Das Ereignis wird zuerst von der Seite ausgelöst, die verlassen werden soll (sofern vorhanden) und dann von der Seite, die aufgerufen werden soll (bevor diese sichtbar wird).

**aPageNew**

Der Parameter enthält den Deskriptor auf das NotebookPage-Objekt, das als nächstes aktiviert wird. Der Deskriptor wird nur übertragen, wenn in aSelecting false übergeben wurde. Kann keine weitere Notizbuchseite aktiviert werden, ist aPageNew = 0.

Die Funktionsdeklaration ohne aPageNew ist ebenfalls möglich.

**Resultat**

## Kontakt

Über den Rückgabewert kann entschieden werden, ob der Vorgang (aufrufen oder verlassen) durchgeführt (true) werden soll oder nicht (false).

Beispiel:

```
sub EvtPageSelect( aEvt : event; // Ereignis aPage : handle; // Notebook-Seite ase
```

## Kontakt

### EvtReadOnlyChanged

#### EditorReadOnly wurde umgesetzt

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aViewId</u>	<u>int</u>	Nummer des Views
<u>aResult</u>	<u>int</u>	Fehlercode
<u>aContentChanged logic</u>		Inhalt wurde zwischenzeitlich geändert
<u>aUserName</u>	<u>alpha</u>	Letzter Benutzer
<u>aTimestamp</u>	<u>caltime</u>	Letzte Änderungszeit
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet

Siehe

#### Ereignisbefehle

Die angegebene Prozedur wird ausgeführt, wenn die Eigenschaft EditorReadOnly des CodeEdit-Objektes geändert wird.

Definition des Funktionskopfes:

```
sub EvtReadOnlyChanged( aEvt : event;      // Ereignis aViewId : int
```

**aEvt**

In aEvt vom Typ event wird unter anderem der Deskriptor des Objekts übergeben, welches das Ereignis ausgelöst hat. Dies ist das Objekt, in dem die Ereignisfunktion eingetragen wurde.

**aViewId**

Hier wird die ID des Views angegeben, in dem die Eigenschaft EditorReadOnly gesetzt oder zurückgesetzt wurde. Wird in mehreren Views das selbe Dokument angezeigt, wird das Ereignis nur für das Hauptview ausgelöst.

**aResult**

Das Resultat gibt an, ob der Modus erfolgreich gewechselt werden konnte. Folgende Werte können zurückgegeben werden:

rOk Wechseln des EditorReadOnly-Status war erfolgreich.

rLocked Dokument ist bereits gesperrt.

rNoRights Keine Rechte vorhanden.

**aContentChanged**

Dieses Argument gibt an, ob das Dokument in der Zwischenzeit verändert wurde. Es wird nur gesetzt, wenn die Eigenschaft EditorReadOnly auf false gesetzt und das Dokument dadurch schreibbar wird. Die Änderungen sind beim Auslösen dieses Ereignisses bereits geladen.

## Kontakt

**aUserName**

**Dieses Argument gibt den Namen des Benutzers an, der das Dokument zuletzt geändert hat.**

**aTimestamp**

**In diesem Argument wird der Zeitstempel der letzten Änderung angegeben.**

**Resultat**

**Der Rückgabewert der Funktion wird nicht ausgewertet.**

**Beispiel:**

```
sub EvtReadOnlyChanged( aEvt : event;      // Ereignis  aViewId : int
```

## Kontakt

### EvtCroNavigate

Ereignis vor Änderung des Inhaltes

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aUrl	alpha	Ziel-URL
Resultat	logic	Gibt an, ob die URL geändert werden soll.
Siehe		<u>Liste, Objekte,</u> <u>Ereignisbefehle</u>

Das Ereignis wird ausgelöst, bevor sich die URL des Inhaltes ändert.

Definition des Funktionskopfes:

```
sub EvtCroNavigate( aEvt : event; // Ereignis aUrl : alpha;
```

**aEvt**

Hier wird der Deskriptor des Chromium-Objektes übergeben.

**aEvt**

Gibt die URL an, zu der navigiert werden soll.

**Resultat**

Der Rückgabewert entscheidet darüber, ob zu der angegebenen URL navigiert werden darf (true) oder nicht (false).

## Kontakt

### Objekte mit EvtCroNavigate

Folgende Objekte haben das Ereignis EvtCroNavigate.

EvtCroNavigate,  
Liste der

Objekte, Liste  
der Ereignisse

- Chromium

Siehe

### Drag & Drop-Ereignisse

#### Dynamisches Drag & Drop

Liste sortiert nach  
Gruppen,  
Alphabetische Liste  
aller Ereignisse,

Siehe Ereignisabläufe Drag & Drop, Drag & Drop-Objekte, Blog,  
Prozedurbeispiel in der Beispiele-Datenbank

- EvtDragInit
- EvtDragTerm
- EvtDrop
- EvtDropEnter
- EvtDropLeave
- EvtDropOver

Bei einer Reihe von Oberflächen-Objekten kann auf Drag & Drop-Operationen in entsprechenden Ereignissen reagiert werden.

Damit diese Objekte das dynamische Drag & Drop unterstützen, muss in der Eigenschaft OleDropMode des Objekts die Ausprägung WinOleDynamic gesetzt werden. Zudem muss angegeben werden, ob das Objekt Quelle, Ziel oder beides von Drag & Drop-Operationen sein soll. Folgende Objekte unterstützen das dynamische Drag & Drop:

- Animation
- AppFrame
- BigIntEdit
- Button
- DataList
- DataListPopup
- DateEdit
- DecimalEdit
- DocView
- Edit
- FloatEdit
- Frame
- GanttGraph
- Icon
- IntEdit
- MdiFrame
- Picture
- RecList
- RecListPopup
- RecView
- TextEdit
- TimeEdit

## Kontakt

- TreeView



Das Objekt RtfEdit verwendet eine eigene Drag & Drop-Steuerung, in die prozedural nicht eingegriffen werden kann.

Bei der Durchführung eines Drag & Drop-Vorganges spielen mehrere Komponenten zusammen. Das Oberflächen-Objekt, das den Vorgang auslöst (z. B. durch Klick in das Objekt) wird DragSource genannt. Das DragSource-Objekt stellt die Datenformate und zugehörigen Dateninhalte bereit, die gezogen werden sollen. Dazu steht ihm das sogenannte DragData-Objekt als Datencontainer zu Verfügung.

Das Oberflächen-Objekt, welches die gezogenen Daten entgegennimmt, wird DropTarget genannt.



Nach dieser Definition kann ein DragSource zugleich auch ein DropTarget sein.

Mit den Tasten und kann die gewünschte Aktion verändert werden. Das Verhalten kann über die Eigenschaften OleDropEffectStandard, OleDropEffectCtrl,

OleDropEffectShift und OleDropEffectCtrlShift des App-Objektes angepasst werden.

Folgende Aktionen werden von CONZEPT 16 standardmäßig durchgeführt:

Tastenkombination Eigenschaft	Aktion
	<u>OleDropEffectStandard</u> <u>_WinDropEffectMove</u>
+	<u>OleDropEffectCtrl</u> <u>_WinDropEffectCopy</u>
+	<u>OleDropEffectShift</u> <u>_WinDropEffectMove</u>
+  +	<u>OleDropEffectCtrlShift</u> <u>_WinDropEffectLink</u>



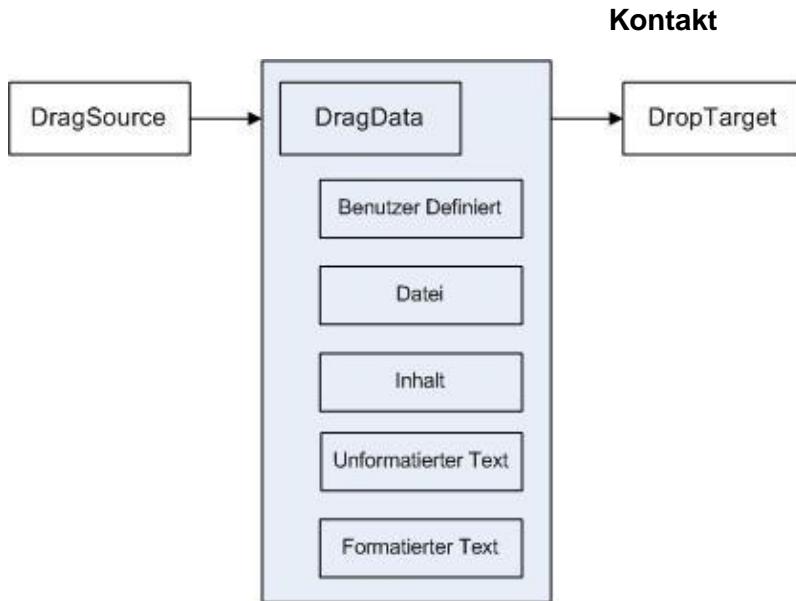
Die gewünschte Aktion wird mit den erlaubten Aktionen aus EvtDragInit und EvtDropEnter verglichen. Ist die gewünschte Aktion bei einem der Ereignisse nicht gesetzt, kann die Drag & Drop-Operation nicht durchgeführt werden.



Damit auf einem Button-Objekt eine Drag & Drop-Operation ausgelöst werden kann, muss die -Taste gedrückt werden.

### EvtDragInit

Ein Drag & Drop-Vorgang wird beim DragSource-Objekt gestartet. Befindet sich das Objekt innerhalb von CONZEPT 16, wird das Ereignis EvtDragInit ausgelöst. Innerhalb dieses Ereignisses werden die an das DropTarget zu übermittelnden Daten aufbereitet. Als Container für die Daten dient das DragData-Objekt. In diesem Objekt werden sowohl das Format der Daten, als auch die Daten selbst abgelegt. In dem Objekt können die Daten auch in unterschiedlichen Formaten angegeben werden. Das DropTarget-Objekt bestimmt, welches der Formate benötigt bzw. ausgewertet werden.



**Das Format der Daten wird in der Eigenschaft FormatEnum angegeben. Dabei wird der Index des Formats übergeben. Soll zum Beispiel formatierter und unformatierter Text übergeben werden müssen zwei Formate gesetzt werden:**

```
aDataObject->wpFormatEnum(_WinDropDataText) # true;aDataObject->wpFormatEnum(_WinDropDataRtf) #
```

**Die eigentlichen Daten werden in einem Data-Objekt angegeben:**

```
tDataText # aDataObject->wpData(_WinDropDataText);tDataRtf # aDataObject->wpData(_WinDropDataRtf)
```

**Die zu übertragenden Daten werden in dem Daten-Objekt übergeben. In diesem Fall handelt es sich um unformatierten und formatierten Text. Diese werden in einem Textpuffer aufbereitet und der Eigenschaft Data zugewiesen.**

```
tText # TextOpen(16); // unformatierten Text aufbereiten... tDataText->wpData # tText;tText # TextO
```

**i** Die Deskriptoren für die Texte oder andere Daten müssen nicht gelöscht werden, sofern die Eigenschaft DataOwner auf true gesetzt ist. Sollen Dateinamen oder benutzerdefinierte Daten übertragen werden, werden diese in Cte-Listen übergeben. Ganze Dateiinhalte können in Memory-Objekten übergeben werden.

Damit die Daten von anderen Programmen ausgewertet werden können, müssen diese in bestimmten Formaten vorliegen. Soll zum Beispiel ein Text in Microsoft Word eingefügt werden, muss der Text formatiert oder unformatiert übergeben werden. Werden beide Texte übergeben, wird der formatierte Text bevorzugt. Soll eine Datei oder mehrere Dateien per Drag & Drop an den Windows Explorer übergeben werden, müssen die Inhalte in Form von Memory-Objekten aufbereitet und in einer Cte-Liste angegeben werden. Befinden sich sowohl das DragSource- als auch das DropTarget-Objekt in einer CONZEPT 16-Applikation, kann das benutzerdefinierte Format (\_WinDropDataUser) verwendet werden.

```
// Übergabe von Dateien an den Explorer// Binäres Objekt öffnen und in ein Memory-Objekt lesenBi
```

Nach der Aufbereitung der Daten muss noch die Operation angegeben werden, die mit diesen Daten möglich sind. Dazu stehen die Konstanten \_WinDropEffectCopy, \_WinDropEffectMove und \_WinDropEffectLink zur Verfügung. Die möglichen

## Kontakt

Operatone werden (auch in Kombination) in aEffect zurückgegeben:

```
aEffect # _WinDropEffectCopy | _WinDropEffectMove;
```

### EvtDropEnter / EvtDropLeave

Diese Ereignisse werden aufgerufen, wenn bei einer Drag & Drop-Operation ein Objekt betreten bzw. verlassen wird. Beim Betreten des Objekts (EvtDropEnter) sollten die gewünschte Operation und die übertragenen Datenformate geprüft werden. In dem Ereignis wird dann definiert, ob die Drag & Drop-Operation durchgeführt werden kann.

```
// Überprüfung des Formatsif (aDataObject->wpFormatEnum(_WinDropDataFile)) aEffect # _WinDropEff
```

Wurde der Vorgang mit einer Operation gestartet, die nicht erlaubt ist, wird das dem Benutzer durch einen entsprechenden Mauszeiger mitgeteilt. Im Ereignis EvtDropLeave müssen nur dann Anweisungen ausgeführt werden, wenn beim EvtDropEnter Objekte geöffnet wurden. In diesem Ereignis können diese Objekte wieder entfernt werden.

### EvtDropOver

Das Ereignis wird während einer Drag & Drop-Operation ausgelöst, wenn der Mauszeiger innerhalb eines Objektes bewegt oder eine der Hilfstasten (, , oder ) betätigt wird.

### EvtDrop

Das Ereignis wird ausgelöst, sobald der Benutzer die Maustaste loslässt und damit eine zugelassene Drag & Drop-Operation innerhalb der CONZEPT 16-Applikation ausführt. Innerhalb des Ereignisses muss nochmals das Format der übergebenen Daten geprüft und das Daten-Objekt ermittelt werden. Anschließend können die Daten ermittelt und verarbeitet werden. Befindet sich das DragSource-Objekt nicht in einer CONZEPT 16-Applikation können die Daten in unterschiedlichen Formaten übergeben werden. Wird zum Beispiel ein Textausschnitt aus Microsoft Word in die Applikation gezogen, liegt der Text in formatierter (als RTF) und als unformatierter Text vor. Der Text wird in einem Textpuffer übergeben. Beim Drag & Drop von externen Dateien aus dem Explorer oder dem Windows Desktop wird eine Liste mit Pfad- und Dateinamen übergeben.

Die unterschiedlichen Formate der zu übertragenen Daten werden in unterschiedlichen Format-Objekten abgebildet. In einem DragData-Objekt können bis zu fünf Format-Objekte angehängt werden.

```
// Überprüfung des Formatsif (aDataObject->wpFormatEnum(_WinDropDataFile)){ tDataFormat # aDatao
```

### EvtDropTerm

Das Ereignis wird bei dem Objekt ausgelöst, in dem der Drag & Drop-Vorgang gestartet wurde, sobald die Operation beendet wird. Das Ereignis wird unabhängig davon, ob die Operation erfolgreich war oder nicht, ausgelöst. Hier können Objekte,

## Kontakt

die in dem Ereignis EvtDragInit angelegt wurde, wieder entfernt werden. Die Objekte, die im DragData-Objekt übergeben werden, müssen nicht entfernt werden.

Eine Beschreibung der beteiligten Objekte beim Drag & Drop befindet sich im Abschnitt Drag & Drop-Objekte. Die Reihenfolge der Ereignisse wird im Abschnitt Ereignisabläufe Drag & Drop erläutert.

## Kontakt

### EvtDragInit

Drag & Drop-Operation starten

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aDataObject</u>	<u>handle DragData-Objekt</u>	
<u>aEffect</u>	<u>int</u>	erlaubte Effekte
<u>aMouseBtn</u>	<u>int</u>	verwendete Maustasten
<u>aDataPlace</u>	<u>handle DataPlace</u>	-Objekt (optional)
<u>Resultat</u>	<u>logic</u>	Drag & Drop-Operation durchführen?

Liste, Objekte, EvtDragTerm,  
Ereignisbefehle,

Siehe

Ereignisabläufe Drag &  
Drop, Blog

Dieses Ereignis wird ausgelöst, wenn die Eigenschaft OleDropMode auf WinOleDynamic gesetzt ist und der Mauszeiger bei gedrückter Maustaste in dem Objekt bewegt wird. In diesem Ereignis werden die Daten für die Drag & Drop-Operation zusammengestellt oder die Operation unterbunden.

Definition des Funktionskopfes:

```
sub EvtDragInit( aEvt : event; // Ereignis aDataObject : handle;
```

**aEvt**

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aDataObject**

In diesem Parameter wird ein Deskriptor auf ein DragData-Objekt übergeben. In diesem Objekt werden die zu übertragenen Daten aufbereitet. Über die Eigenschaften dieses Objekts werden auch die Formate angegeben, die übertragen werden können.

**aEffect**

Hier werden die möglichen Drag & Drop-Operationen zurückgegeben. Folgende Konstanten können dazu verwendet werden:

WinDropEffectNone Drag & Drop-Operation abbrechen

WinDropEffectCopy Daten dürfen kopiert werden

WinDropEffectMove Daten dürfen verschoben werden

WinDropEffectLink Daten dürfen verlinkt werden

Die Konstanten WinDropEffectCopy, WinDropEffectMove und WinDropEffectLink können miteinander verknüpft werden.

## Kontakt

### aMouseBtn

Die zum Starten der Drag & Drop-Operation gedrückte Maustaste wird in diesem Parameter übergeben. Der Wert kann mit folgenden Konstanten verglichen werden:

<u>WinMouseLeft</u>	linke Maustaste
<u>WinMouseRight</u>	rechte Maustaste
<u>WinMouseMiddle</u>	mittlere Maustaste

Die Maustaste kann mit den Werten WinMouseShift und / oder WinMouseCtrl kombiniert sein.

 Damit eine Drag & Drop-Operation auf einem Button-Objekt ausgelöst werden kann, muss die  -Taste gedrückt werden.

### aDataPlace

Hier wird der Deskriptor auf ein DataPlace-Objekt übergeben. Aus diesem Objekt können zusätzliche Informationen über die Position der übertragenen Daten im Quell-Objekt ermittelt werden.

### Resultat

Um die Drag & Drop-Operation durchzuführen muss true zurückgegeben und in aEffect ein Wert ungleich WinDropEffectNone übergeben werden.

Wird eine Drag & Drop-Operation durch die Rückgabe von false verhindert, wird anschließend das Ereignis EvtMouse bzw. EvtMouseItem ausgelöst.

### Beispiel:

In diesem Beispiel werden die Daten eines binären Objekts so aufbereitet, dass sie in den Windows Explorer oder auf den Desktop gezogen werden können.

```
sub EvtDragInit( aEvt          : event;      // Ereignis  aDataObject : handle;    // DragData-Objekt
```

Die in dem Beispiel angelegten Objekte (Cte-Liste und Memory-Objekt) müssen in dem Ereignis EvtDragTerm wieder entfernt werden.

## Kontakt

### EvtDragTerm

Drag & Drop-Operation beenden

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aDataObject	handle DragData-Objekt	
aEffect	int	durchgeführte Operation

Resultat logic Wird nicht ausgewertet

Liste, Objekte, EvtDragInit, Ereignisbefehle,

Siehe Ereignisabläufe Drag & Drop, Blog

Dieses Ereignis wird nur ausgelöst, wenn die Eigenschaft OleDropMode auf WinOleDynamic gesetzt ist und eine Drag & Drop-Operation abgeschlossen wurde. In diesem Ereignis können angelegte Objekte, die nicht durch das System entfernt werden, gelöscht werden.

Definition des Funktionskopfes:

```
sub EvtDragTerm( aEvt : event; // Ereignis aDataObject : handle;
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aDataObject

In diesem Parameter wird der Deskriptor auf das DragData-Objekt übergeben. Das Objekt wird beim Starten der Drag & Drop-Operation mit Informationen gefüllt. Wird die Operation aus CONZEPT 16 heraus gestartet, werden die Daten im Ereignis EvtDragInit zur Verfügung gestellt.

aEffect

Hier wird die durchgeführte Drag & Drop-Operation übergeben. Der Wert kann mit folgenden Konstanten verglichen werden:

WinDropEffectNone Drag & Drop-Operation abgebrochen

WinDropEffectCopy Daten wurden kopiert

WinDropEffectMove Daten wurden verschoben

WinDropEffectLink Daten wurden verknüpft

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

**EvtDropEnter**  
Objekt betreten

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aDataObject</u>	<u>handle DragData-Objekt</u>	
<u>aEffect</u>	<u>int</u>	erlaubte Effekte
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet
		<u>Liste, Objekte,</u>
		<u>EvtDropLeave,</u>
<u>Siehe</u>		<u>Ereignisbefehle,</u>
		<u>Ereignisabläufe Drag &amp;</u>
		<u>Drop, Blog</u>

Dieses Ereignis wird ausgelöst, wenn die Eigenschaft OleDropMode auf WinOleDynamic gesetzt ist und das Objekt während einer Drag & Drop-Operation betreten wird. Beim Start der Drag & Drop-Operation wird das Ereignis ebenfalls ausgelöst.

Definition des Funktionskopfes:

```
sub EvtDropEnter( aEvt : event; // Ereignis aDataObject : handle;
```

**aEvt**

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**aDataObject**

In diesem Parameter wird der Deskriptor auf das DragData-Objekt übergeben. Das Objekt wird beim Starten der Drag & Drop-Operation mit Informationen gefüllt. Wird die Operation aus CONZEPT 16 heraus gestartet, werden die Daten im Ereignis EvtDragInit zur Verfügung gestellt.

**aEffect**

Hier werden die möglichen Drag & Drop-Operationen zurückgegeben. Folgende Konstanten können dazu verwendet werden:

- WinDropEffectNone Drag & Drop-Operation nicht möglich
- WinDropEffectCopy Daten dürfen kopiert werden
- WinDropEffectMove Daten dürfen verschoben werden
- WinDropEffectLink Daten dürfen verlinkt werden

Die Konstanten WinDropEffectCopy, WinDropEffectMove und WinDropEffectLink können miteinander verknüpft werden. Wird eine Verschiebe-Operation durchgeführt, aber in dieser Variablen nur eine Kopieraktion zugelassen, wird dem Benutzer durch einen veränderten Mauszeiger darauf hingewiesen, dass die Funktion nicht ausgeführt werden kann.

## Kontakt

### Resultat

**Der Rückgabewert der Funktion wird nicht ausgewertet.**

### Beispiel:

```
sub EvtDropEnter( aEvt : event; // Ereignis aDataObject : handle;
```

## Kontakt

### EvtDropLeave

Objekt verlassen

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen <b>Wird nicht</b>

Resultat logic

ausgewertet

Liste, Objekte,

EvtDropEnter,

Siehe Ereignisbefehle,

Ereignisabläufe Drag &

Drop, Blog

Dieses Ereignis wird ausgelöst, wenn die Eigenschaft OleDropMode auf WinOleDynamic gesetzt ist und der Mauszeiger das Objekt verlässt. In diesem Ereignis können Objekte, die im EvtDropEnter angelegt wurden, wieder entfernt werden.

**Definition des Funktionskopfes:**

```
sub EvtDropLeave( aEvt : event; // Ereignis) : logic;
```

**aEvt**

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

**Resultat**

Der Rückgabewert der Funktion wird nicht ausgewertet.

**Beispiel:**

```
sub EvtDropLeave( aEvt : event; // Ereignis) : logic;
```

## Kontakt

### EvtDropOver

In Objekt bewegen / Hilfstaste drücken

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aDataObject	handle DragData-Objekt	
aDataPlace	handle DataPlace-Objekt	

		Eingabe: vom
aEffect	int	Benutzer gewählter Effekt, Ausgabe: durchgeführter Effekt
aMouseBtn	int	Verwendete Maustasten
Resultat	logic	Wird nicht ausgewertet
		<u>Liste, Objekte, Ereignisbefehle,</u>

Siehe

Ereignisabläufe Drag & Drop, Blog

Dieses Ereignis wird ausgelöst, wenn während einer Drag & Drop-Operation das Ereignis EvtDropEnter ausgelöst wurde und anschließend die Maus in dem Objekt

bewegt wird oder eine der Hilfstasten (, , ) betätigt bzw. losgelassen wird. Beim Start der Drag & Drop-Operation wird das Ereignis ebenfalls für das Drag-Objekt ausgelöst, wenn dieses als Drop-Ziel erlaubt ist (OleDropMode != WinOleSource).

Definition des Funktionskopfes:

```
sub EvtDropOver( aEvt : event; // Ereignis aDataObject : handle;
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aDataObject

In diesem Parameter wird der Deskriptor auf das DragData-Objekt übergeben. Das Objekt wird beim Starten der Drag & Drop-Operation mit Informationen gefüllt. Wird die Operation aus CONZEPT 16 heraus gestartet, werden die Daten im Ereignis EvtDragInit zur Verfügung gestellt.

aDataPlace

Hier wird der Deskriptor auf ein DataPlace-Objekt übergeben. Aus diesem Objekt können zusätzliche Informationen über die Position der übertragenen Daten ermittelt werden.

aEffect

## Kontakt

In diesem Parameter wird die gewünschte Operation übergeben. Die Operation wird beim Starten des Drag & Drop-Vorgangs definiert. Wird die Operation in CONZEPT 16 gestartet, erfolgt die Definition in dem Ereignis EvtDragInit. Die durchzuführende Operation muss in die Variable aEffect geschrieben werden. Der Wert wird an das Ereignis EvtDragTerm übergeben und kann dort entsprechend ausgewertet werden. Der Wert muss nur dann gesetzt werden, wenn die durchzuführende Operation von dem übergebenen Wert abweicht.

Der Wert kann mit den gleichen Konstanten verglichen und gesetzt werden.

WinDropEffectNone Drag & Drop-Operation abbrechen / Operation wurde abgebrochen

WinDropEffectCopy Daten sollen kopiert werden / wurden kopiert

WinDropEffectMove Daten sollen verschoben werden / wurden verschoben

WinDropEffectLink Daten sollen verlinkt werden / wurden verlinkt Wird der Effekt übergeben, können die Konstanten WinDropEffectCopy,

WinDropEffectMove und WinDropEffectLink miteinander verknüpft sein. Beim Setzen des Effekts sollte nur eine der Konstanten angegeben werden.

Effekte, die durch EvtDropEnter bereits ausgeschlossen wurden, können hier nicht wieder zugelassen werden.

aMouseBtn

Die zum Starten der Drag & Drop-Operation gedrückte Maustaste wird in diesem Parameter übergeben. Der Wert kann mit folgenden Konstanten verglichen werden:

WinMouseLeft linke Maustaste

WinMouseRight rechte Maustaste

WinMouseMiddle mittlere Maustaste

Die Maustaste kann mit den Werten WinMouseShift und / oder WinMouseCtrl kombiniert sein.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.



Ausführliche Beispiele zum Thema Drag & Drop befinden sich in der CodeLibrary-Datenbank, die aus dem Kundenbereich auf unserer Webseite ([www.vectorsoft.de](http://www.vectorsoft.de)) heruntergeladen werden kann.

## Kontakt

### EvtDrop

#### Drag & Drop-Operation ausführen

Name	Typ	Beschreibung
aEvt	event	Ereignisinformationen
aDataObject	handle	DragData-Objekt
aDataPlace	handle	DataPlace-Objekt
aEffect	int	Operation
aMouseBtn	int	verwendete Maustaste
Resultat	logic	Wird nicht ausgewertet
		<u>Liste, Objekte, Ereignisbefehle,</u>

Siehe

Ereignisabläufe Drag & Drop, Blog

Dieses Ereignis wird ausgelöst, wenn die Eigenschaft OleDropMode auf WinOleDynamic gesetzt ist und die Maustaste mit einer Drag & Drop-Operation über dem Objekt losgelassen wurde.



Damit dieses Ereignis ausgelöst wird, muss der jeweilige Effekt im EvtDropEnter erlaubt werden. EvtDropEnter wird auch beim Start der Drag & Drop-Operation ausgelöst, um festzustellen, ob EvtDrop im Zielobjekt erlaubt ist.

Definition des Funktionskopfes:

```
sub EvtDrop( aEvt : event;      // Ereignis aDataObject : handle;    // D
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aDataObject

In diesem Parameter wird der Deskriptor auf das DragData-Objekt übergeben. Das Objekt wird beim Starten der Drag & Drop-Operation mit Informationen gefüllt. Wird die Operation aus CONZEPT 16 heraus gestartet, werden die Daten im Ereignis EvtDragInit zur Verfügung gestellt.

aDataPlace

Hier wird der Deskriptor auf ein DataPlace-Objekt übergeben. Aus diesem Objekt können zusätzliche Informationen über die Position der übertragenen Daten ermittelt werden.

aEffect

In diesem Parameter wird die gewünschte Operation übergeben. Die Operation wird beim Starten des Drag & Drop-Vorgangs definiert. Wird die Operation in CONZEPT 16 gestartet, erfolgt die Definition in dem Ereignis EvtDragInit. Die durchgeführte

## Kontakt

Operation muss in die Variable aEffect geschrieben werden. Der Wert wird an das Ereignis EvtDragTerm übergeben und kann dort entsprechend ausgewertet werden. Der Wert muss nur dann gesetzt werden, wenn die durchgeführte Operation von dem übergebenen Wert abweicht.

Der Wert kann mit den gleichen Konstanten verglichen und gesetzt werden.

WinDropEffectNone Drag & Drop-Operation abbrechen / Operation wurde abgebrochen

WinDropEffectCopy Daten sollen kopiert werden / wurden kopiert

WinDropEffectMove Daten sollen verschoben werden / wurden verschoben

WinDropEffectLink Daten sollen verlinkt werden / wurden verlinkt Wird der Effekt übergeben, können die Konstanten WinDropEffectCopy,

WinDropEffectMove und WinDropEffectLink miteinander verknüpft sein. Beim Setzen des Effekts sollte nur eine der Konstanten angegeben werden.

Die Überprüfung, ob die Daten in diesem Objekt fallen gelassen werden dürfen, muss bereits im Ereignis EvtDropEnter erfolgen.

aMouseBtn

Die zum Starten der Drag & Drop-Operation gedrückte Maustaste wird in diesem Parameter übergeben. Der Wert kann mit folgenden Konstanten verglichen werden:

WinMouseLeft linke Maustaste

WinMouseRight rechte Maustaste

WinMouseMiddle mittlere Maustaste

Die Maustaste kann mit den Werten WinMouseShift und / oder WinMouseCtrl kombiniert sein.

Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

Beispiel

```
sub EvtDrop( aEvt : event;      // Ereignis   aDataObject : handle;      // Datencontainer
            case aDataObject->wpFormatEnum(_WinDropDataText) : {           // unformatierter Text wurde üb
```

 Ausführliche Beispiele zum Thema Drag & Drop befinden sich in der CodeLibrary-Datenbank, die aus dem Kundenbereich auf unserer Webseite ([www.vectorsoft.de](http://www.vectorsoft.de)) heruntergeladen werden kann.

## Kontakt

**Mehrfachselektions-Ereignisse**

Ereignisse, die bei der Mehrfachselektion auftreten können Liste

sortiert nach

Gruppen,

Alphabetische Liste

aller Ereignisse,

Siehe Objekte der

Mehrfachselektion,

Prozedur-Beispiel in

der

Beispiele-Datenbank

- EvtLstSelectRange

## Kontakt

### EvtLstSelectRange

#### Selektion eines Bereichs

Name	Typ	Beschreibung
<u>aEvt</u>	<u>event</u>	Ereignisinformationen
<u>aSelDataObj</u>	<u>handle</u>	<b>Deskriptor des SelectionData-Objekts</b>
<u>aID1</u>	<u>int / bigint</u>	Erster Datensatz der Selektion
<u>aID2</u>	<u>int / bigint</u>	Letzter Datensatz der Selektion
<u>aDirection</u>	<u>int</u>	Richtung der Selektion
<u>Resultat</u>	<u>logic</u>	Wird nicht ausgewertet <u>Liste, Objekte,</u>

Siehe

#### Ereignisbefehle

Dieses Ereignis wird ausgelöst, wenn in dem Objekt eine Mehrfachselektion mit gedrückter  -Taste ausgeführt wird.

Definition des Funktionskopfes:

```
sub EvtLstSelectRange( aEvt : event; // Ereignis aSelDataObj : hand
```

Alternativer Funktionskopf:

```
sub EvtLstSelectRange( aEvt : event; // Ereignis aSelDataObj : hand
```

aEvt

In diesem Parameter vom Typ event wird unter anderem der Deskriptor des auslösenden Objekts, d. h. das Objekt, in dem die Ereignisfunktion eingetragen wurde, übergeben.

aSelDataObj

In diesem Übergabeparameter wird das SelectionData-Objekt übergeben. Über das Objekt können alle bisher selektierten Datensätze ermittelt und die zu selektierenden Datensätze hinzugefügt werden.

aId1 und aId2

In diesen Parametern wird die Datensatz-ID des ersten und des letzten Datensatzes angegeben, die selektiert wurden. Der Programmierer kann durch Lesen der Datensätze und der Schlüsselinformation aus dem auslösenden RecList-Objekt alle dazwischenliegenden Datensätze ermitteln.



Wird der alternative Funktionskopf verwendet, kommt es zum Laufzeitfehler ErrValueOverflow, wenn die zu übergebende Datensatz-ID nicht in den Wertebereich von int passt.

aDirection

## Kontakt

Hier wird die Richtung, in der die Datensätze selektiert wurden übergeben. Der Parameter kann folgende Werte annehmen:

**WinMsdSelDown** Die Datensätze sind von oben nach unten in der Liste selektiert worden.

**WinMsdSelUp** Die Datensätze sind von unten nach oben in der Liste selektiert worden.

## Resultat

Der Rückgabewert der Funktion wird nicht ausgewertet.

### Beispiel:

```
sub EvtLstSelectRange( aEvt : event; // Ereignis aSelDataObj : handle; // SelectD  
else { // Dateinummer ermitteln tFileNo # aEvt:Obj->wpDbFileNo; // Schlüsselnummer bz
```

 Es ist zu beachten, dass möglicherweise nicht alle Datensätze, die in der Liste selektiert sind, in die Liste der Datensätze übernommen werden. Die Datensätze, die zwischenzeitlich gelöscht wurden, werden nicht übernommen. Wird der letzte Datensatz der Selektion gelöscht, terminiert die Schleife nicht rechtzeitig. Ein Löschen kann durch das Sperren des Datensatzes verhindert werden.

## Kontakt

### Ablaufpläne Ereignisse

Ablaufpläne zu Ereignissen aus der CONZEPT 16-Oberfläche Siehe  
Ereignisse

In diesem Kapitel befinden sich Ablaufpläne zu den Ereignissen aus der CONZEPT 16 Oberfläche. Dieses Kapitel soll dem Entwickler als Referenz für Reihenfolge und Verhalten der CONZEPT 16-Ereignisverarbeitung dienen. Folgende Ablaufpläne können diesem Kapitel entnommen werden.

- Ereignisabläufe Drag & Drop
- Ereignisabläufe MdiFrame
- Ereignisabläufe RecList
- Ereignisabläufe RecView
- Ereignisabläufe WinLstEdit

## Kontakt

### Ereignisabläufe Drag & Drop

#### Ablaufpläne zur Ereignisverarbeitung bei Drag & Drop

Drag &  
Drop-Objekte,  
Drag &

Siehe Drop-Ereignisse,  
Ablaufpläne  
Ereignisse

#### Ablauf der Ereignisse

Wird beispielsweise die linke Maustaste in einem Oberflächen-Objekt gedrückt und gleichzeitig um einen kleinen Bereich bewegt, löst die DragSource das Ereignis EvtDragInit aus.

In diesem Ereignis kann die DragSource entscheiden, ob ein Drag & Drop-Vorgang gestartet werden soll oder nicht und gegebenenfalls die entsprechenden Daten in dem DragData-Objekt aufbereiten. Wird keine Drag & Drop-Operation durchgeführt, erfolgt nach dem Ereignis EvtDragInit der Aufruf des Ereignisses EvtMouse bzw. EvtMenuItem.

Wird der Vorgang gestartet, erhält ein DropTarget-Objekt zunächst ein Ereignis, wenn der Mauszeiger das DropTarget-Objekt betritt (EvtDropEnter). Dieses kann dann entscheiden, ob die bereitgestellten Daten des DragData-Objekts unterstützt werden oder nicht. Durch den Rückgabewert wird vom System der dazugehörige Mauszeiger angezeigt.

Es können folgende Fälle auftreten:

1. Die Daten werden durch das DropTarget-Objekt unterstützt und der Benutzer lässt die Daten fallen

Es wird das Ereignis EvtDrop für das DropTarget-Objekt ausgelöst. In dem Ereignis können die Daten aufbereitet und in das Objekt übernommen werden.

2. Die Daten werden durch das DropTarget-Objekt nicht unterstützt und der Benutzer lässt die Daten fallen

Werden die Daten in diesem Objekt fallen gelassen oder der Vorgang abgebrochen, wird kein Ereignis aufgerufen.

3. Der Benutzer bewegt die Maus aus dem Objekt hinaus

Es wird das Ereignis EvtDropLeave für das DropTarget-Objekt ausgelöst.

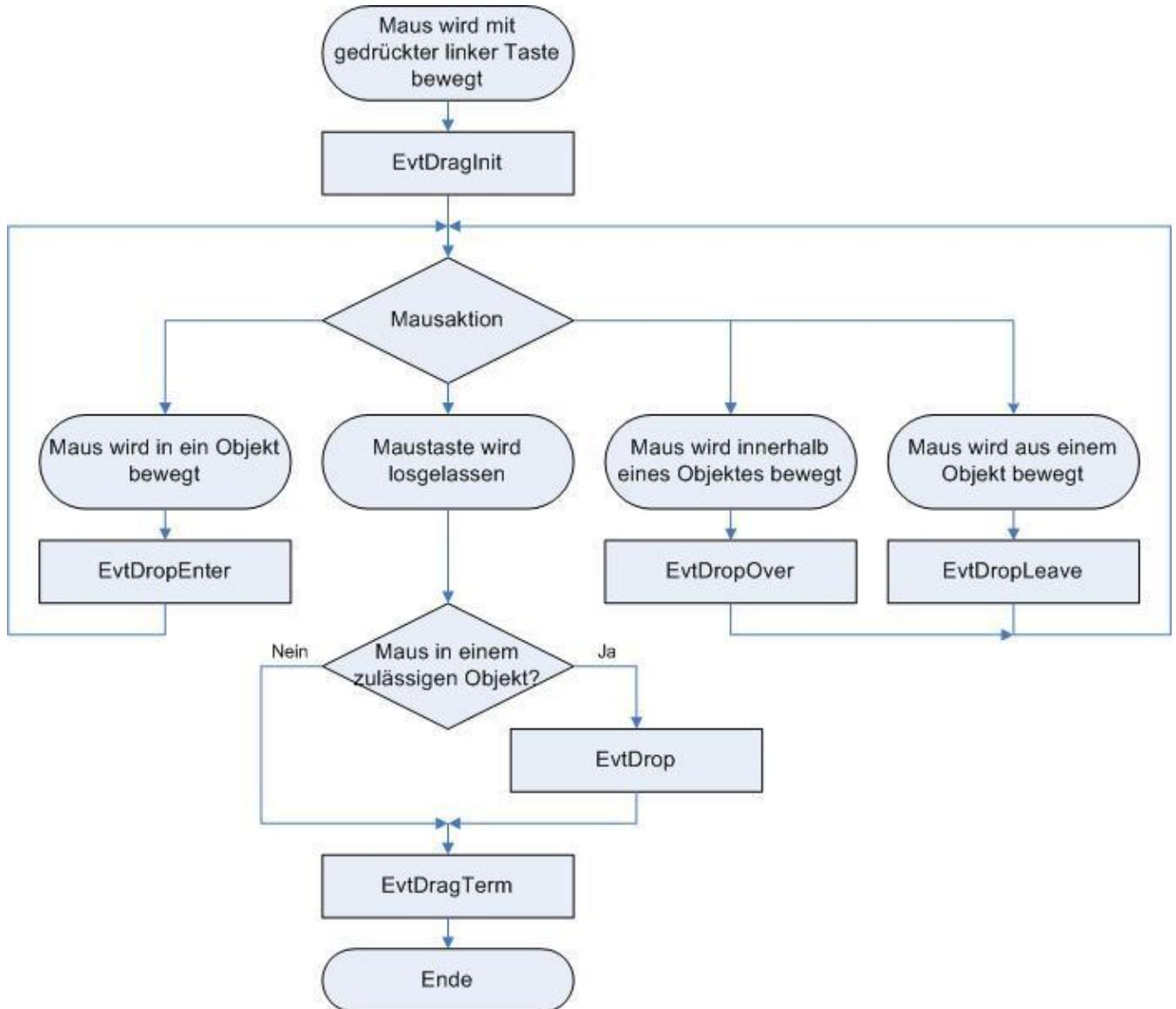
4. Der Benutzer bewegt die Maus in einem Objekt

Es wird das Ereignis EvtDropOver für das DropTarget-Objekt ausgelöst.

Im Fall a) wird also EvtDrop aufgerufen. Im Fall c) wird EvtDropLeave aufgerufen und der Drag & Drop-Vorgang läuft weiter. Es wird wieder ein Ereignis erzeugt, wenn der Benutzer die Maus in das DropTarget bewegt usw. Im Fall d) wird EvtDropOver aufgerufen und der Drag & Drop-Vorgang läuft ebenfalls weiter.

## Kontakt

Im Fall a) und b) ist der Drag & Drop-Vorgang beendet. Das DragSource-Objekt erhält die Benachrichtigung EvtDragTerm. Im Fall a) mit positivem Ausgang ("Daten wurden übertragen") und im Fall b) mit negativem Ausgang ("Keine Daten wurden übertragen"). Die DragSource erhält also in jedem Fall eine Benachrichtigung, wenn der Drag & Drop-Vorgang beendet wurde.



Die Ereignisse EvtDragInit und EvtDragTerm werden nur dann ausgelöst, wenn sich das DragSource-Objekt in der CONZEPT 16-Applikation befindet. Ebenso werden die Ereignisse EvtDropEnter, EvtDropOver, EvtDropLeave und EvtDrop nur ausgelöst, wenn das DropTarget-Objekt sich in der CONZEPT 16-Applikation befindet.

## Kontakt

Ereignisabläufe MdiFrame

Ablaufpläne zur Ereignisverarbeitung eines MdiFrames

MdiFrame,

Ereignisse eines

Siehe MdiFrame-Objekts,

Ablaufpläne

Ereignisse

- Neues MDI-Fenster anlegen

- MDI-Fenster wechseln

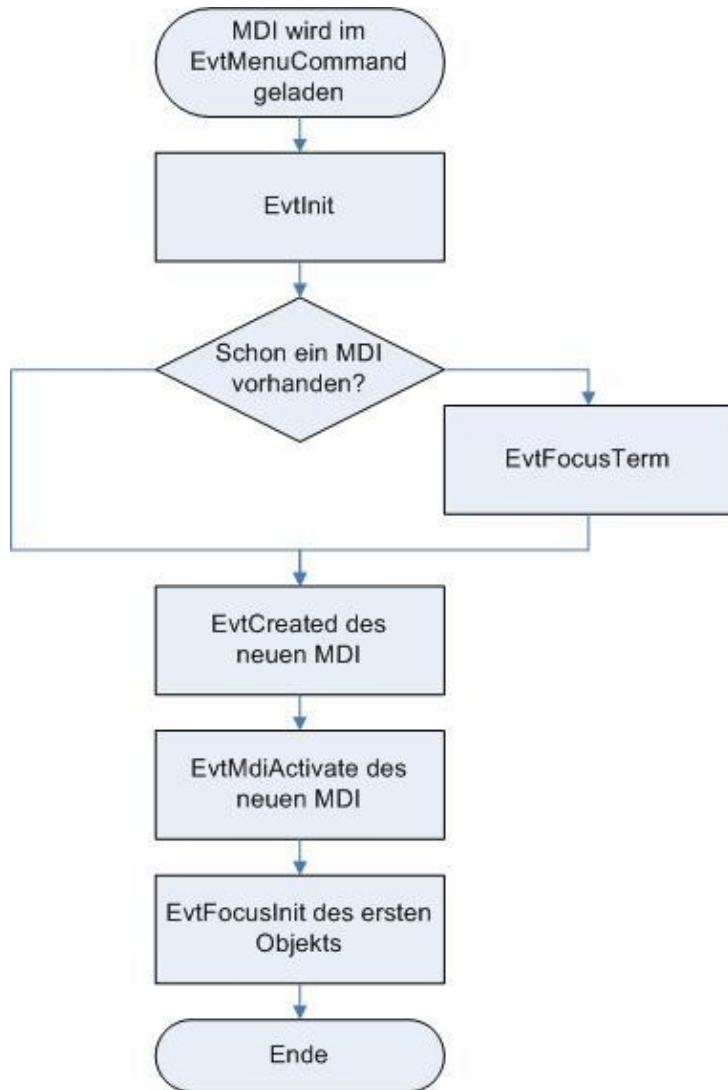
- MDI-Fenster schließen

Neues MDI-Fenster anlegen

Ein neues MDI-Frame wird in der Regel über einen Menüpunkt angelegt. Dabei wird entweder die Anweisung WinAdd() oder WinAddByName() verwendet. Beim Laden des Fensters wird das Ereignis EvtInit des MDI-Fensters durchgeführt.

Wird bereits ein anderes MDI-Fenster angezeigt, wird für das Objekt, das dort den Eingabefokus besitzt ein EvtFocusTerm ausgeführt. Anschließend folgen die Ereignisse EvtMdiActivate für das neu geladenen MDI-Fenster und EvtFocusInit für das erste Objekt innerhalb des Fensters, das den Fokus bekommen kann.

## Kontakt

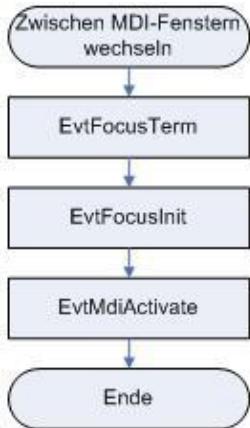


### MDI-Fenster wechseln

Wird zwischen zwei MDI-Fenstern gewechselt wird zunächst das EvtFocusTerm für das Objekt, das den Fokus in dem Fenster besitzt, aus dem gewechselt werden soll, durchgeführt. Anschließend wird das Ereignis EvtMdiActivate für das MDI-Fenster aufgerufen, das betreten werden und EvtFocusInit für das Objekt, das dort den Fokus bekommen soll.

Bei diesen Ereignissen ist bereits der Datenbereich des zu aktivierenden Fensters instanziert.

## Kontakt

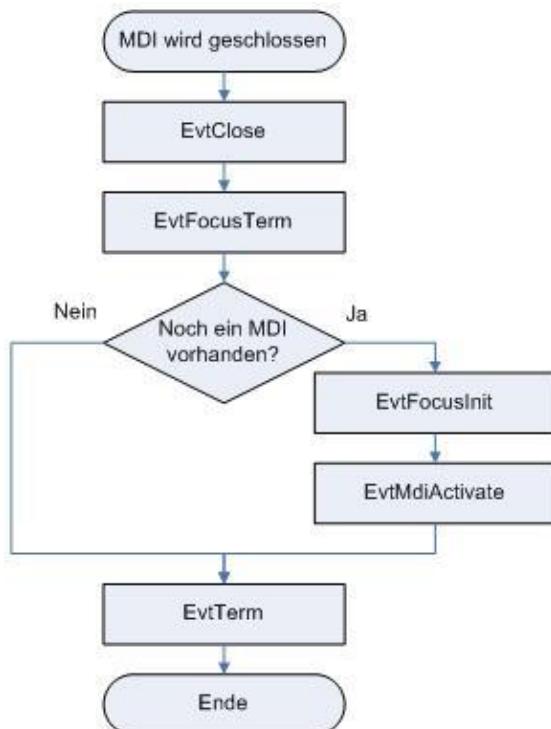


### MDI-Fenster schließen

Beim Schließen eines MDI-Fensters wird zunächst dessen EvtClose aufgerufen. Wird das Fenster tatsächlich geschlossen, wird das Ereignis EvtFocusTerm des Objekts aufgerufen, das zur Zeit den Fokus besitzt.

Sind noch weitere MDI-Fenster vorhanden, wird das nächste aktiviert. Dabei werden die Ereignisse EvtFocusInit und EvtMdiActivate aufgerufen. In allen drei Ereignissen steht ein evtl. zugewiesener Datenbereich (siehe DbVar) des zu schließenden Fensters nicht mehr zur Verfügung. Bei einem Fensterwechsel ist bereits im Ereignis EvtFocusTerm der Datenbereich des neuen Fenstes instanziert.

Zum Schluss wird das Ereignis EvtTerm des zu schließenden Fensters ausgeführt.



## Kontakt

### Ereignisabläufe RecList

#### Ablaufpläne zur Ereignisverarbeitung einer RecList

RecList,  
Ereignisse eines  
Siehe RecList-Objekts,  
Ablaufpläne  
Ereignisse

- Laden eines RecList-Objekts
- Das RecList-Objekt erhält den Focus
- Das RecList-Objekt wird aktualisiert

#### Laden eines RecList-Objekts

Nach dem Laden eines Frame-Objekts werden alle enthaltenen Objekte erzeugt. Sobald das RecList-Objekt erzeugt wurde, wird das Ereignis EvtInit der RecList ausgelöst.

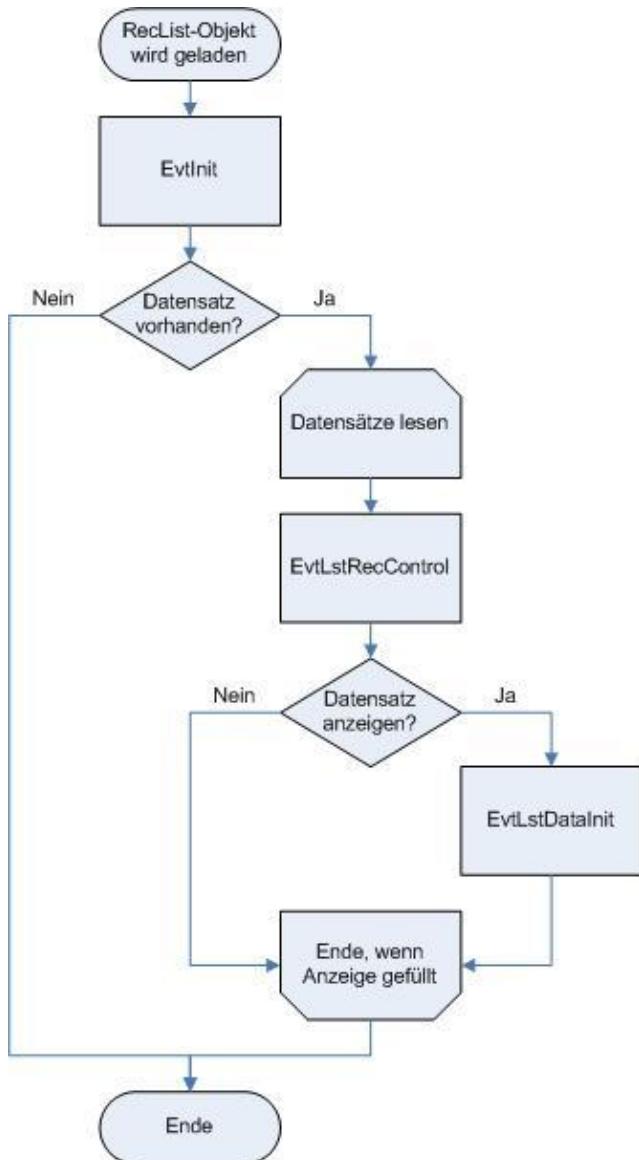
Sind in der zugrundeliegenden Datei (Eigenschaften DbFileNo und DbKeyNo) Datensätze vorhanden, werden solange Datensätze gelesen, bis der Anzeigebereich mit Datensätzen gefüllt oder keine weiteren Datensätze gelesen werden können.

Für jeden gelesenen Datensatz wird das Ereignis EvtLstRecControl durchgeführt. Dieses Ereignis entscheidet, ob der Datensatz in der Liste angezeigt wird oder nicht.

Wird ein Datensatz angezeigt, wird das Ereignis EvtLstDataInit aufgerufen. Hier können letzte Änderungen vor der Anzeige des Datensatzes durchgeführt werden.

Die Ereignisse EvtLstRecControl und EvtLstDataInit werden solange durchgeführt, bis der Anzeigebereich der Liste gefüllt oder das Ende der Datei erreicht ist.

## Kontakt

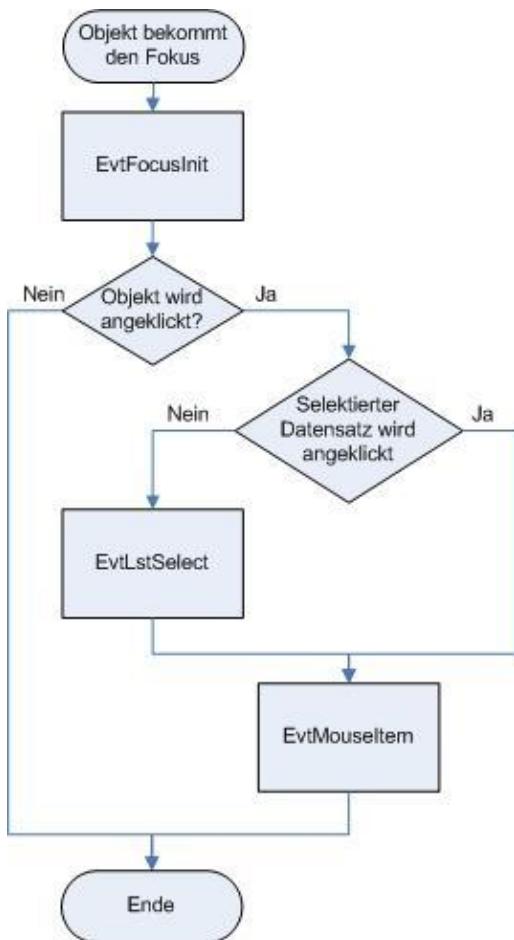


**Das RecList-Objekt erhält den Fokus**

**Das RecList-Objekt kann entweder durch Drücken der -Taste oder durch Anklicken mit der Maus den Fokus bekommen. In beiden Fällen wird das Ereignis EvtFocusInit durchgeführt. Nur wenn mit der Maus das Objekt ausgewählt wurde, werden weitere Ereignisse aufgerufen.**

**Wurde mit der Maus auf den selektierten Datensatz in der Liste geklickt, wird das Ereignis EvtMouseItem ausgelöst. Wurde ein nicht selektierter Datensatz angeklickt, wird zuvor noch das Ereignis EvtLstSelect ausgeführt.**

## Kontakt



**Das RecList-Objekt wird aktualisiert**

**Das RecList-Objekt wird durch den Befehl WinUpdate() aktualisiert:**

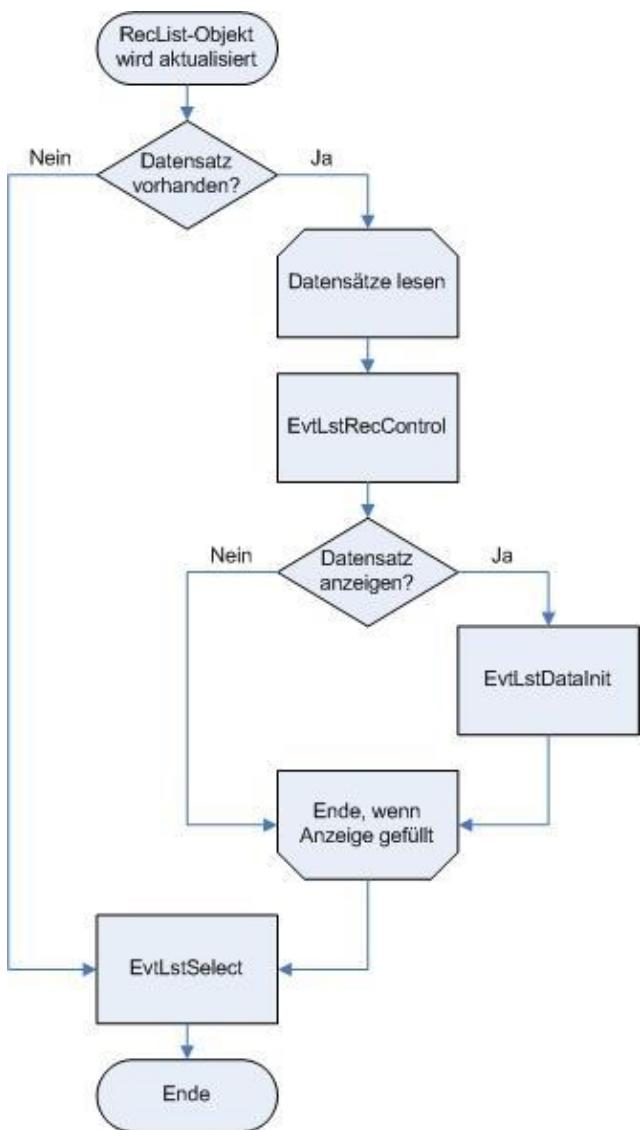
```
tHd1RecList->WinUpdate(_WinUpdOn, _WinLstRecFromBuffer | _WinLstPosSelected | _WinLstRecDoSelect)
```

Unabhängig davon, ob der Datensatz bereits selektiert ist oder nicht, wird der Anzeigebereich der Liste neu aufgebaut. Das heißt, dass die Datensätze, die im Anzeigebereich dargestellt werden, die Ereignisse EvtLstRecControl und EvtLstDataInit durchgeführt werden, bis entweder der Anzeigebereich gefüllt oder das Ende der Datei erreicht ist. Zum Schluss wird das Ereignis EvtLstSelect für den Datensatz in den Feldpuffern ausgeführt.



**Das EvtLstSelect wird auch dann ausgeführt, wenn die Liste keine Datensätze enthält.**

## Kontakt



## Kontakt

### Ereignisabläufe RecView

#### Ablaufpläne zur Ereignisverarbeitung einer RecView

RecView,  
Ereignisse eines  
Siehe RecView-Objekts,  
Ablaufpläne  
Ereignisse

- Laden eines RecView-Objektes
- Das RecView-Objekt erhält den Focus
- Das RecView-Objekt wird aktualisiert
- Items des RecView-Objektes werden editiert.

#### Laden eines RecView-Objektes

Nach dem Laden eines Frame-Objekts werden alle enthaltenen Objekte erzeugt. Sobald das RecView-Objekt erzeugt wurde, wird das Ereignis EvtInit des RecView ausgelöst.

Sind in der zugrundeliegenden Datei (Eigenschaft DbFileNo und DbKeyNo) Datensätze vorhanden, werden solange Datensätze gelesen, bis der Anzeigebereich mit Datensätzen gefüllt oder keine weiteren Datensätze gelesen werden können.

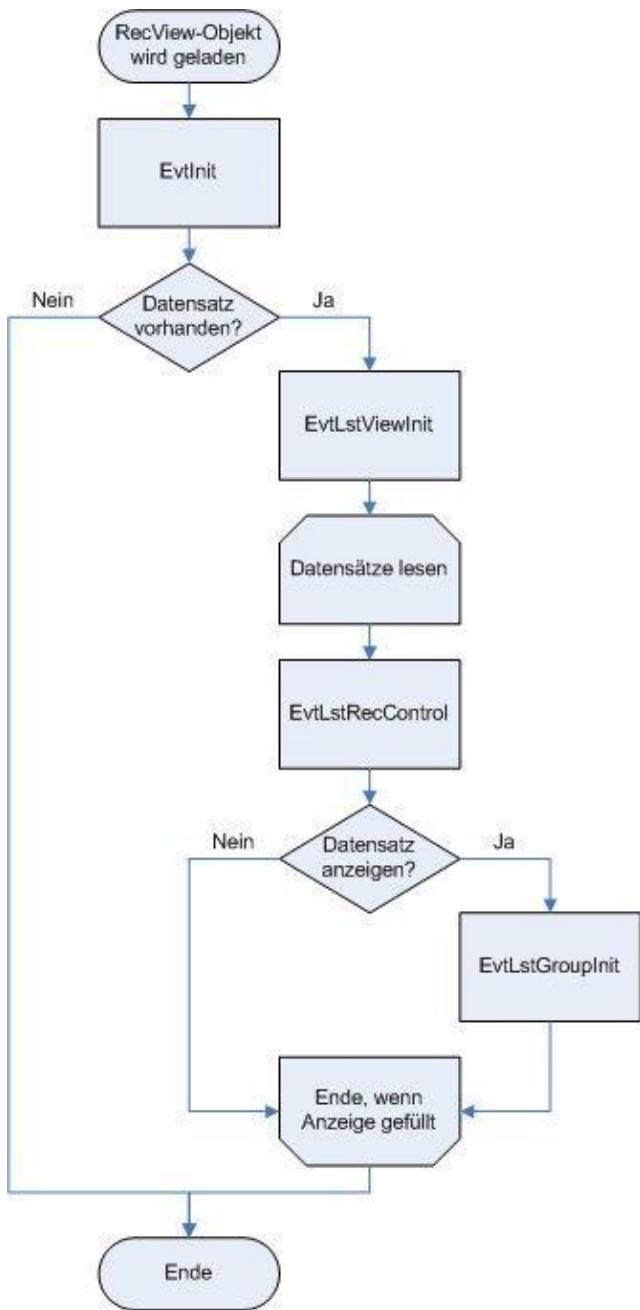
Bei der ersten Anzeige eines Views wird das Ereignis EvtLstViewInit ausgelöst, wenn Datensätze zur Anzeige vorhanden sind.

Für jeden gelesenen Datensatz wird das Ereignis EvtLstRecControl durchgeführt. Dieses Ereignis entscheidet, ob der Datensatz in der Liste angezeigt wird oder nicht.

Wird ein Datensatz angezeigt, wird das Ereignis EvtLstGroupInit aufgerufen. Hier können letzte Änderungen vor der Anzeige des Datensatzes durchgeführt werden.

Die Ereignisse EvtLstRecControl und EvtLstGroupInit werden solange durchgeführt, bis der Anzeigebereich des RecView gefüllt oder das Ende der Datei erreicht ist.

## Kontakt

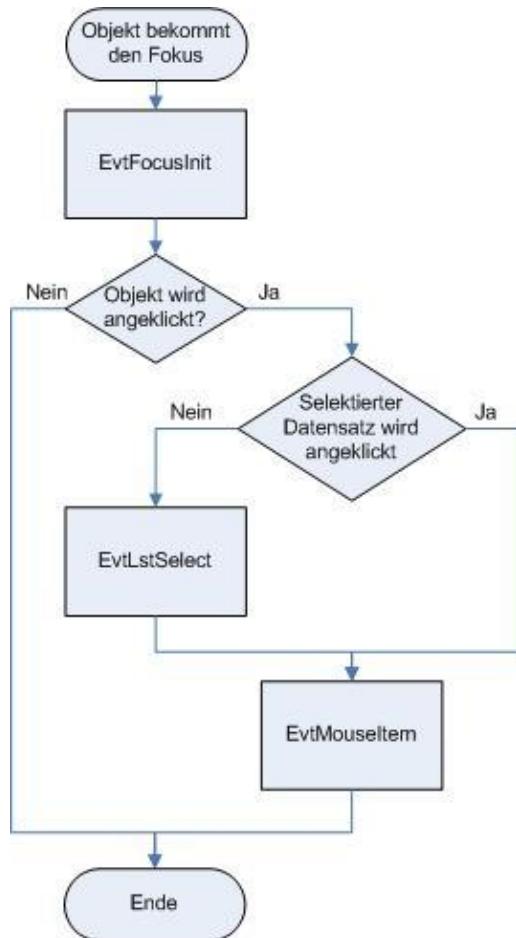


**Das RecView-Objekt erhält den Fokus**

**Das RecView-Objekt kann entweder durch Drücken der -Taste oder durch Anklicken mit der Maus den Fokus bekommen. In beiden Fällen wird das Ereignis EvtFocusInit durchgeführt. Nur wenn mit der Maus das Objekt ausgewählt wurde, werden weitere Ereignisse aufgerufen.**

**Wurde mit der Maus auf den selektierten Datensatz in der Liste geklickt, wird das Ereignis EvtMouseItem ausgelöst. Wurde ein nicht selektierter Datensatz angeklickt, wird zuvor noch das Ereignis EvtLstSelect ausgeführt.**

## Kontakt



Das RecView-Objekt wird aktualisiert

Das RecView-Objekt wird durch den Befehl WinRvwUpdate() aktualisiert:

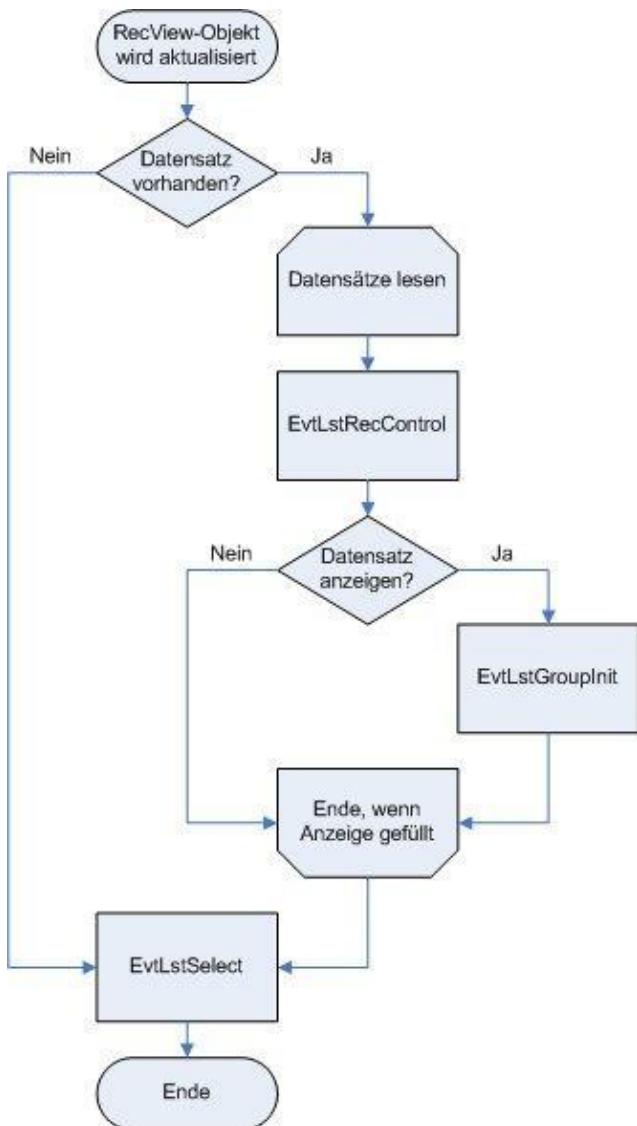
```
tHdlRecView->WinRvwUpdate(_WinRvwUpdateFromSelected | _WinRvwUpdateDoSelect, 0);
```

Unabhängig davon, ob der Datensatz bereits selektiert ist oder nicht, wird der Anzeigebereich der Liste neu aufgebaut. Das heißt, dass die Datensätze, die im Anzeigebereich dargestellt werden, die Ereignisse EvtLstRecControl und EvtLstGroupInit durchgeführt werden, bis entweder der Anzeigebereich gefüllt oder das Ende der Datei erreicht ist. Zum Schluss wird das Ereignis EvtLstSelect für den Datensatz in den Feldpuffern ausgeführt.



Das EvtLstSelect wird auch dann ausgeführt, wenn die Liste keine Datensätze enthält.

## Kontakt



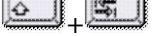
Items des RecView-Objektes werden editiert

Durch die in EditKeyStart und EditMouseStart definierten Tastatur- und Mausaktionen wird das Ereignis EvtLstEditStartGroup ausgelöst. In diesem Ereignis kann der Datensatz gesperrt werden. Über den Parameter aResult wird bestimmt, ob der Datensatz bearbeitet werden darf, oder nicht. Soll er bearbeitet werden, wird als nächstes das Ereignis EvtLstEditStartItem aufgerufen. In diesem kann entschieden werden, ob das angeklickte bearbeitet werden darf. Ist dies der Fall, kann im Ereignis EvtLstEditActivate das Eingabeobjekt gefüllt werden. Bei RTF-Texten ist dies zwingend notwendig.

Wird das Eingabeobjekt verlassen, die - oder die in EditKeyEnd definierte Taste gedrückt, wird das Ereignis EvtLstEditEndItem ausgelöst. In diesem Ereignis kann der Inhalt des Eingabeobjektes in die Feldpuffer übertragen werden. Im anschließenden Ereignis EvtLstEditEndGroup kann der Datensatz gespeichert oder entsperrend gelesen werden. In beiden Ereignissen wird per Parameter aAbort übergeben, ob die

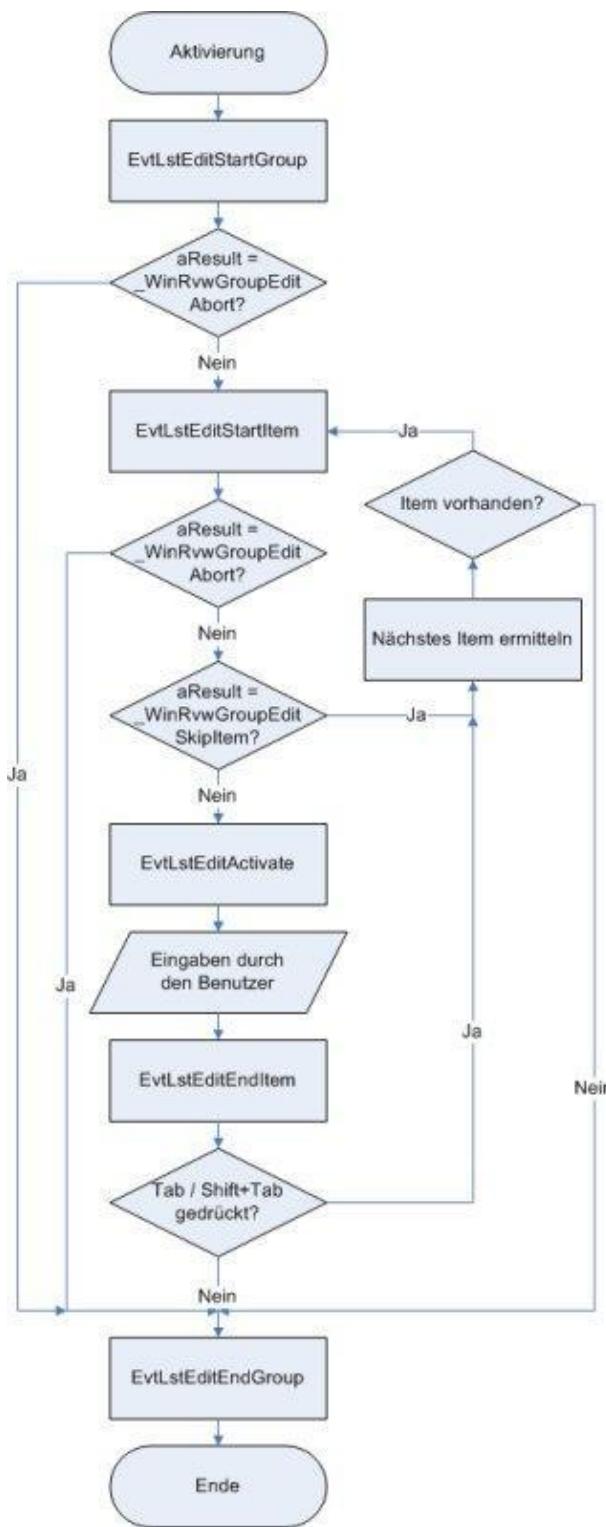
Aktion abgebrochen werden soll. Wird das Eingabeobjekt hingegen mit oder

## Kontakt

 verlassen, wird das nächste oder vorherige sichtbare (Visible = true) und editierbare (ReadOnly = false) Item gesucht und erneut das Ereignis EvtLstEditStartItem aufgerufen.

Das Ereignis EvtLstEditEndItem wird auch aufgerufen, wenn die Bearbeitung in dem Ereignis EvtLstEditStartItem abgebrochen wurde. Wird die Bearbeitung bereits im Ereignis EvtLstEditStartGroup abgebrochen, so wird das Ereignis EvtLstEditEndGroup ausgelöst.

## Kontakt



## Kontakt

Ereignisabläufe WinLstEdit()

Ablaufpläne zur Ereignisverarbeitung beim Editieren einer Listenspalte

WinLstEdit(),

Siehe Ablaufpläne

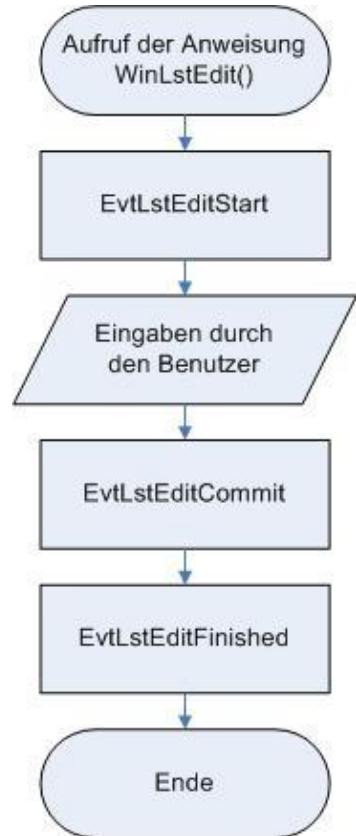
Ereignisse

Spalte einer Liste editieren

Der Aufruf des Befehls WinLstEdit() löst das Ereignis EvtLstEditStart aus. In diesem Ereignis kann eine erzeugte ComboBox mit entsprechenden Daten gefüllt werden.

Wird das Eingabeobjekt verlassen (entweder mit einer Taste / Tastenkombination oder durch das Anklicken eines anderen Oberflächen-Objekts mit der Maus), wird das Ereignis EvtLstEditCommit aufgerufen. In diesem Ereignis kann bestimmt werden, ob die Eingabe in die Liste übernommen werden soll.

Anschließend wird das Ereignis EvtLstEditFinished aufgerufen. Hier kann entschieden werden, ob ein weiterer Eintrag in der Liste editiert werden soll. Die Taste oder Mausaktion, die zum Verlassen des Eingabeobjektes geführt hat, wird an das Ereignis übergeben.



## Ereignisse

In dieser Liste sind alle Ereignisse aufgeführt.

### Liste sortiert

Siehe

### Gruppen,

- [EvtAdviseDDE](#)
- [EvtAttachState](#)
- [EvtChanged](#)
- [EvtChangedActive](#)
- [EvtChangedChild](#)
- [EvtChangedDesign](#)
- [EvtClicked](#)
- [EvtClose](#)
- [EvtCreated](#)
- [EvtCroNavigate](#)
- [EvtCtxEvent](#)
- [EvtDbFldUpdate](#)
- [EvtDragInit](#)
- [EvtDragTerm](#)
- [EvtDrop](#)
- [EvtDropEnter](#)
- [EvtDropLeave](#)
- [EvtDropOver](#)
- [EvtEndSession](#)
- [EvtFocusCancel](#)
- [EvtFocusInit](#)
- [EvtFocusTerm](#)
- [EvtFsiMonitor](#)
- [EvtHelpTip](#)
- [EvtHyphenate](#)
- [EvtInit](#)
- [EvtIvlDropItem](#)
- [EvtJob](#)
- [EvtKeyItem](#)
- [EvtLstDataInit](#)
- [EvtLstEditActivate](#)
- [EvtLstEditCommit](#)
- [EvtLstEditEndGroup](#)
- [EvtLstEditEndItem](#)
- [EvtLstEditFinished](#)
- [EvtLstEditStart](#)
- [EvtLstEditStartGroup](#)
- [EvtLstEditStartItem](#)
- [EvtLstGroupArrange](#)
- [EvtLstGroupInit](#)
- [EvtLstRecControl](#)
- [EvtLstSelect](#)
- [EvtLstSelectRange](#)
- [EvtLstViewInit](#)

## Kontakt

- EvtMdiActivate
- EvtMenuCommand
- EvtMenuContext
- EvtMenuInitPopup
- EvtMenuPopup
- EvtMouse
- EvtMenuItem
- EvtMouseMove
- EvtNodeExpand
- EvtNodeSearch
- EvtNodeSelect
- EvtPageSelect
- EvtPosChanged
- EvtReadOnlyChanged
- EvtSocket
- EvtTapi
- EvtTerm
- EvtTimer
- EvtUser

## Kontakt

### Programmierbeispiele

#### Beispiele zur Programmierung mit CONZEPT 16

Dieses Kapitel enthält Beispiele zur Programmierung mit CONZEPT 16. Alle Beispiele gehören thematisch zu einem entsprechenden Dokumentationsabschnitt. Weitere Beispiele können der CONZEPT 16-Beispiele-Datenbank entnommen werden. Folgende Beispiele befinden sich in diesem Kapitel:

- [Beispiel - Datei herunterladen](#)
- [Beispiel - Drucken einer Liste](#)
- [Beispiel - Durchsuchen einer XML-Struktur](#)
- [Beispiel - Editieren in einem RecList-Objekt](#)
- [Beispiel - Eigenschaft Custom](#)
- [Beispiel - Einbinden einer DLL](#)
- [Beispiel - EvtInit / EvtTerm](#)
- [Beispiel - HLS-Farbwerte](#)
- [Beispiel - JSON-Datei](#)
- [Beispiel - PDF-Dokument drucken](#)
- [Beispiel - RGB-Farbwerte](#)
- [Beispiel - Schreiben von externen Dateien / Asynchroner Dialog](#)
- [Beispiel - Selektionen](#)
- [Beispiel - Selektionszeitpunkt](#)
- [Beispiel - Sortierung einer Zeichenkette](#)
- [Beispiel - switch / EvtMenuItem / EvtLstDataInit](#)
- [Beispiel - UserInfo / EvtMenuInitPopup](#)
- [Beispiel - Verknüpfungen](#)
- [Beispiel - Verwendung der COM-Schnittstelle](#)
- [Beispiel - Web-Anbindung](#)

## Kontakt

**Beispiel - Datei herunterladen (über HTTP-Objekt)**

Einfache Kommunikation über HTTP zum Download einer Datei Siehe  
[HTTP](#)

In diesem Beispiel wird mit Hilfe des [HTTP](#)-Objekts eine Datei von einem HTTP-Server heruntergeladen und lokal gespeichert. Beim Aufruf der Funktion muss die URI und der Pfad und Dateiname angegeben werden.

```
sub DownloadFile( aURI : alpha(512); aDestFile : alpha(); int; lo
try { // Verbindung aufbauen tSck # SckConnect(tHost, 80, 0, 10000); // Anfrage erzeu
```

## Kontakt

### Beispiel - Drucken einer Liste

#### Eine einfache Liste auf den Standard-Drucker ausgeben Befehle

Siehe [zum Drucken](#).

#### Beispiel ohne Beschreibung

In diesem Beispiel wird eine einfache Liste in der Druckvorschau angezeigt. Aus der Druckvorschau kann diese Liste anschließend auf den Drucker ausgegeben werden. Voreingestellt ist der Windows-Standarddrucker.

Bei diesem Beispiel wird davon ausgegangen, dass ein PrintForm-Objekt angelegt wurde, das Ausgabeobjekte (zum Beispiel PrtText) enthält, die mit der Datenstruktur verbunden sind.

Zur Ausgabe der Seitennummern wird ebenfalls ein PrintForm-Objekt verwendet, das nur aus einem PrtText-Objekt besteht. In das PrtText-Objekt wird die Seitennummer geschrieben.

Zu Beginn der Prozedur werden verschiedene Variablen angelegt, die im weiteren Verlauf der Prozedur benötigt werden.

```
@A+@C+main local { tJob : int; // Job-Deskriptor tForm : int; // PrintFo
```

Zunächst wird ein temporärer Printjob geöffnet. Dies erfolgt mit dem Befehl PrtJobOpen(). Der Ausdruck wird nur dann fortgesetzt, wenn der PrintJob geöffnet werden konnte.

```
tJob # PrtJobOpen(_PrtDocDinA4, '', _PrtJobOpenWrite | _PrtJobOpenTemp); if (tJob > 0) {
```

Im Anschluss werden die PrintForm-Objekte mit dem Befehl PrtFormOpen() geladen.

Im Druckelement für die Adresse sind die Position und die Formatierung der Feldinhalte eines Datensatzes festgelegt. Die PrintForm kann mit unterschiedlichen Datensätzen immer wieder gedruckt werden, ohne neu geladen werden zu müssen. Bei komplexeren Listen können auch mehrere PrintForm-Objekte geladen sein und im Wechsel ausgedruckt werden.

Die Verarbeitung wird auch hier nur dann fortgesetzt, wenn beide PrintForm-Objekte geladen werden konnten.

```
tForm # PrtFormOpen(_PrtTypePrintForm, 'Liste.Adresse'); tFormFooter # PrtFormOpen(_PrtTyp
```

Da der Deskriptor des Objektes, dass die Seitennummer aufnimmt, häufiger verwendet wird, wird dieser in der Variablen tPageNoObj abgelegt.

```
tPageNoObj # tFormFooter->PrtSearch('PrtText');
```

Jetzt wird die erste Seite zum Drucken angelegt. Dies erfolgt mit dem Befehl PrtJobWrite(). Mit diesem Befehl werden ebenfalls Seitenumbrüche und die letzte Seite geschrieben. Die Verarbeitung wird unterbrochen, wenn diese Seite nicht angelegt werden konnte.

```
tPage # tJob->PrtJobWrite(_PrtJobPageStart); if (tPage > 0) {
```

## Kontakt

Da die Seitenzahl unabhängig vom verwendeten Seitenformat am Ende der Seite gedruckt werden soll, wird die Höhe des druckbaren Bereiches ermittelt und die Höhe des PrintForm-Objektes mit der Seitennummer davon abgezogen. Das entspricht der vertikalen Position am Ende der Seite.

```
tMaxSize # tPage->ppBoundMax;           tPageNoPos # tMaxSize:y - tFormFooter->ppFormHeight;
```

Zu diesem Zeitpunkt ist alles für den eigentlichen Ausdruck vorbereitet. In einer Schleife werden jetzt alle Datensätze der Datei fKunden gelesen (RecRead()).

```
tErr # RecRead(fKunden, kKndID, _RecFirst);           while (tErr < _rNoRec) {
```

Innerhalb der Schleife wird zunächst geprüft, ob das PrintForm-Element noch vollständig auf die Seite gedruckt werden kann. Hierzu wird der bereits verbrauchte Platz (Eigenschaft BoundAdd) und der noch zur Verfügung stehende Platz (Eigenschaft BoundMax) der Seite ermittelt.

Überschreitet der verbrauchte Platz plus die Höhe des zu druckenden Elements plus die Höhe der Fußzeile mit der Seitenzahl (Eigenschaft FormHeight) die maximal zur Verfügung stehende Höhe der Seite, wird die Fußzeile an eine bestimmte Seitenposition gedruckt und mit PrtJobWrite() mit der Option PrtJobPageBreak ein Seitenumbruch eingefügt. Anschließend wird die Seitenzahl um eins erhöht.

Kann keine neue Seite begonnen werden, wird die Schleife abgebrochen.

```
tAddSize # tPage->ppBoundAdd;           if (tAddSize:y + tForm->ppFormHeight + tFormFoot
```

In der Schleife wird jeder Datensatz gelesen und über das Anfügen der Form mit dem Befehl PrtAdd() in den Druckjob geschrieben.

```
tPage->PrtAdd(tForm);           tErr # RecRead(fKunden, kKndID, _RecNext); }
```

Nachdem die Schleife beendet ist, wird die Seitennummer der letzten Seite gedruckt und die Seite mit dem Befehl PrtJobWrite() mit der Option PrtJobPageEnd abgeschlossen.

```
tPageNoObj->ppCaption # ' - ' + CnvAI(tJob->PrtInfo(_PrtJobPageCount) + 1) + ' - ';
```

Das PrintForm-Objekt wird nicht weiter benötigt und mit dem Befehl PrtFormClose() entladen.

```
tForm->PrtFormClose();           tFormFooter->PrtFormClose(); }
```

Mit dem Befehl PrtJobClose() wird die Druckaufbereitung abgeschlossen, der Druckjob beendet und die Druckvorschau aufgerufen.

```
tErr # tJob->PrtJobClose(_PrtJobPreview); }
```

Die Liste wird nicht sofort nach dem Beenden der Druckaufbereitung ausgedruckt. In der Druckvorschau können der Drucker und unterschiedliche Einstellungen zu dem betreffenden Drucker (zum Beispiel Schachtauswahl) vorgenommen werden.

## Kontakt

**Beispiel - Durchsuchen einer XML-Struktur**

**Alle Knoten einer Baum-Struktur durchlaufen**

Befehle für  
dynamische

Siehe Strukturen, Befehle  
für  
XML-Verarbeitung

In diesem Beispiel wird ein Algorithmus zum Durchlaufen einer Baumstruktur aus CteNode-Objekten vorgestellt. Mit der Funktion können alle Knoten eines gelesenen XML-Dokuments ermittelt werden.

Die Funktion wird rekursiv aufgerufen und startet mit dem Wurzelknoten des Baumes.

```
sub XmlRead( aXmlNode : handle; ) local{ tValueAlpha : alpha(4096); // Inhalt des Knotens tXmlNode
```

Die Funktion kann nach dem Einlesen eines XML-Dokuments zur Anzeige in einem TreeView-Objekt aufgerufen werden.

```
main local{ tErr      : int;  tXmlDoc : handle; }{ // Wurzel-Knoten erzeugen tXmlDoc # CteOpen(_Ct
```

## Kontakt

**Beispiel - Editieren in einem RecList-Objekt**

**Eintrag in einem RecList-Objekt ändern**

WinLstEdit(),  
EvtLstEditStart,  
**Siehe** EvtLstEditCommit,  
EvtLstEditFinished

In diesem Beispiel wird gezeigt, wie ein Eintrag in einem RecList-Objekt verändert werden kann.

**Das Beispiel besteht aus fünf Funktionen:**

- main

In dieser Funktion wird der verwendete Dialog angezeigt.

- sub EvtMouseItem

Das Anklicken eines Datensatzes führt zum Editieren der angeklickten Spalte.

- sub EvtLstEditStart

Der entsprechende Datensatz wird gesperrt.

- sub EvtLstEditCommit

Die Änderungen am Datensatz werden übernommen oder verworfen.

- sub EvtLstEditFinished

Der Datensatz wird gespeichert und entsperrt.

**main**

In dieser Funktion wird lediglich das Fenster des Beispiels geladen und angezeigt. Das Fenster beinhaltet nur ein RecList-Objekt. Dem Objekt sind in den Ereignissen EvtMouseItem, EvtLstEditStart, EvtLstEditCommit und EvtLstEditFinished die entsprechenden Ereignis-Funktionen zugeordnet.

```
@A+@C+main{ WinDialog('RecListEdit');}
```

**sub EvtMouseItem**

In diesem Ereignis wird der Editiervorgang gestartet.

```
sub EvtMouseItem( aEvt : event; // Ereignis aButton : int; // Maustaste aHitTest : int; )
```

Zunächst wird überprüft, ob mit der Maus ein Datensatz doppelt angeklickt wurde. Damit die Funktion später einfacher erweitert werden kann, erfolgt die Prüfung innerhalb eines switch...case...default-Konstruktions.

```
switch (true) { case ((aButton = _WinMouseLeft | _WinMouseDouble) and (aHitTest = _W...)) { ... } }
```

Mit dem Befehl WinLstEdit() wird der Editiervorgang gestartet. Der Befehl benötigt den Deskriptor des RecList-Objekts (aEvt:Obj) und der angeklickten Spalte (aItem).

```
WinLstEdit(aEvt:Obj, aItem); }
```

## Kontakt

Der Befehl WinLstEdit() löst das Ereignis EvtLstEditStart aus.

sub EvtLstEditStart

In diesem Ereignis werden die Vorbereitungen zum Editieren getroffen. In diesem Beispiel wird dabei lediglich der aktuell selektierte Datensatz in der Liste gelesen und gesperrt. Hier können aber auch ComboBoxen mit Inhalten gefüllt werden.

```
sub EvtLstEditStart( aEvt : event; // Ereignis aColumn : int; // Spalte aEdit
```

Die Datensatz-ID des aktuell in der Liste selektierten Datensatzes ist in der Eigenschaft DbRecID des RecList-Objekts enthalten. In dem Übergabeparameter aEvt:Obj wird der Deskriptor des RecList-Objekts übergeben. Die notwendigen Informationen zum Lesen des Datensatzes können so aus den Eigenschaften der Liste entnommen werden.

```
RecRead(aEvt:Obj->wpDbFileNo, 0, _RecID | _RecLock, aEvt:Obj->wpDbRecID); return(true);
```

Der Deskriptor des Eingabeobjektes wird in dem Parameter (aEdit) übergeben. Mit Hilfe dieses Deskriptors können die Eigenschaften des Eingabeobjektes (Farben, Standardwert usw.) verändert werden.

sub EvtLstEditCommit

Das Ereignis EvtLstEditCommit wird aufgerufen, sobald der Eingabefokus das Eingabeobjekt verlässt. Die Taste, mit der das Objekt verlassen wurde, wird als Parameter (aKey) der Funktion übergeben. Der Rückgabewert der Funktion entscheidet darüber, ob der Inhalt des Eingabeobjektes in die Liste übertragen wird. In diesem Beispiel wird der Inhalt des Eingabeobjektes nur dann nicht in die Liste

übertragen, wenn der Benutzer das Objekt mit der Taste  verlässt.

```
sub EvtLstEditCommit( aEvt : event; // Ereignis aColumn : int; // Spalte aKey
```

sub EvtLstEditFinished

Nach dem Ereignis EvtLstEditCommit wird das Ereignis EvtLstEditFinished aufgerufen. Hier kann der Editiermodus abgeschlossen oder eine weitere Spalte der Liste editiert werden. In diesem Beispiel wird der gesperrte Datensatz zurückgeschrieben und der Editiermodus beendet.

```
sub EvtLstEditFinished( aEvt : event; // Ereignis aColumn : int; // Spalte
```

Unabhängig davon, ob Änderungen am Datensatz vorgenommen wurden, wird der Datensatz zurückgeschrieben. Die notwendigen Informationen werden aus den Eigenschaften der Liste gelesen. Anschließend wird die Liste aktualisiert, um den geänderten Datensatz anzuzeigen.

```
tErg # RecReplace(aEvt:Obj->wpDbFileNo, _RecUnlock); aEvt:Obj->WinUpdate(_WinUpdOn, _WinLstFro
```

Soll durch einfaches Anklicken einer Spalte mit einem logischen Wert dieser geändert werden, kann dazu nicht der Befehl WinLstEdit() verwendet werden. Der Befehl würde

## Kontakt

ein Eingabeobjekt mit einer ComboBox erzeugen. Statt dessen kann in dem Ereignis EvtMouseItem der einfache Klick abgefangen und die notwendigen Operationen zum Ändern des Datensatzes durchgeführt werden.

## Kontakt

### Beispiel - Editieren in einem RecView-Objekt

Eintrag in einem RecView-Objekt ändern

EvtLstEditStartGroup,  
EvtLstEditStartItem,

Siehe EvtLstEditActivate,  
EvtLstEditEndItem,  
EvtLstEditEndGroup

In diesem Beispiel wird gezeigt, wie ein Eintrag in einem RecView-Objekt verändert werden kann.

Das Beispiel besteht aus sechs Funktionen:

- main

In dieser Funktion wird der verwendete Dialog angezeigt.

- sub EvtLstEditStartGroup

Der entsprechende Datensatz wird gesperrt. Die Bearbeitung der Gruppe kann unterbunden werden.

- sub EvtLstEditStartItem

Das Bearbeiten einzelner Items kann unterbunden werden.

- sub EvtLstEditActivate

Der Inhalt des Items wird in das Editierobjekt übertragen.

- sub EvtLstEditEndItem

Der Inhalt des des Editierobjekts kann in die Feldpuffer übertragen werden.

- sub EvtLstEditEndGroup

Der Datensatz wird gespeichert und entsperrt.

#### Funktion main

In dieser Funktion wird lediglich das Fenster des Beispiels geladen und angezeigt. Das Fenster beinhaltet nur ein RecView-Objekt. Dem Objekt sind in den Ereignissen EvtLstEditStartGroup, EvtLstEditStartItem, EvtLstEditActivate, EvtLstEditEndItem und EvtLstEditEndGroup die entsprechenden Ereignis-Funktionen zugeordnet.

Der Editiervorgang wird durch Doppelklick auf ein Item gestartet.

```
@A+@C+main{ WinDialog('RecViewEdit'); }
```

#### sub EvtLstEditStartGroup

In diesem Ereignis werden die Vorbereitungen zum Editieren getroffen. In diesem Beispiel wird dabei lediglich der aktuell selektierte Datensatz in dem RecView gelesen und gesperrt.

```
sub EvtLstEditStartGroup( aEvt : event; // Ereignis aGroup : h
```

## Kontakt

Der Datensatzpuffer des aktuellen Datensatzes wird in dem Parameter aRecBuf übergeben. Die notwendigen Informationen zum Lesen und Sperren des Datensatzes können so aus diesem Parameter entnommen werden.

```
aRecBuf->RecRead(0, _RecID | _RecLock | _RecNoLoad, aRecBuf->RecInfo(_RecID));}
```

Wird im Parameter aResult die Konstante WinRvwGroupEditAbort gesetzt, wird die Bearbeitung abgebrochen und das Ereignis EvtLstEditEndGroup ausgelöst. Andernfalls kommt als nächstes das Ereignis EvtLstEditStartItem.

sub EvtLstEditStartItem

In diesem Ereignis kann die Bearbeitung einzelner Items unterbunden werden. In diesem Beispiel wird dies bei einem Item, das ein Bild anzeigt, gemacht.

```
sub EvtLstEditStartItem( aEvt : event; // Ereignis aGroup : han
```

Im Parameter aGroup sind die Eigenschaften SelectorItem und SelectorSubItem auf das aktuelle Item gesetzt. Über diese Eigenschaften kann entschieden werden, ob die Bearbeitung fortgesetzt werden darf.

```
if (aGroup->wpSelectorItem = 1 and aGroup->wpSelectorSubItem = 3) aResult # _WinRvwGroupEditAb
```

Wird im Parameter aResult die Konstante WinRvwGroupEditAbort gesetzt, wird die Bearbeitung abgebrochen und das Ereignis EvtLstEditEndItem ausgelöst. Andernfalls kommt als nächstes das Ereignis EvtLstEditActivate.

sub EvtLstEditActivate

Das Ereignis nach dem Ereignis EvtLstEditStartItem ausgelöst. In diesem Ereignis kann man den Inhalt des Editierobjektes anpassen. Bei der Bearbeitung eines Rtf-Textes ist das RtfEdit leer und muss in diesem Ereignis beispielsweise mit dem Befehl WinRtfLoad() gefüllt werden.

```
sub EvtLstEditActivate( aEvt : event; // Ereignis aGroup : han
```

sub EvtLstEditEndItem

Das Ereignis wird aufgerufen, wenn der Editievorgang für das aktuelle Item oder SubItem beendet oder abgebrochen wird. In diesem Ereignis kann man den geänderten Wert in den Datensatzpuffer übertragen.

```
sub EvtLstEditEndItem( aEvt : event; // Ereignis aGroup : han
```

Im Parameter aAbort wird übergeben, ob die Bearbeitung abgebrochen wurde, oder sie mit der -Taste beendet wurde.

```
if (!aAbort and aGroup->wpSelectorItem = 1 and aGroup->wpSelectorSubItem = 4) { // Rtf-Text i
```

Nach diesem Ereignis wird EvtLstEditEndGroup ausgeführt.

sub EvtLstEditEndGroup

## Kontakt

Das Ereignis wird aufgerufen, wenn der Editiervorgang beendet oder abgebrochen wird. In diesem Beispiel wird in dem Ereignis der Datensatz gesichert und entsperrt.

```
sub EvtLstEditEndGroup( aEvt : event; // Ereignis aGroup : han
```

Im Parameter aAbort wird übergeben, ob die Bearbeitung in einem vorangegangenen Ereignis mit WinRvwGroupEditAbort abgebrochen wurde.

```
// nur Abbruch -> Datensatz entsperren if (aAbort) aRecBuf->RecRead(0, _RecID | _RecUnlock |
```

Nach diesem Ereignis ist die Bearbeitung beendet.

## Kontakt

### Beispiel - Eigenschaft Custom

Mehrere Werte in der Custom-Eigenschaft speichern Siehe  
Custom

In diesem Beispiel werden zwei Funktionen vorgestellt, mit denen mehrere Werte in der Eigenschaft Custom gesetzt und abgefragt werden können.

Die einzelnen Werte werden ähnlich wie in HTML-Tags in der Eigenschaft eingetragen. Sie bestehen aus einem Namen und einem Wert. Der Name wird durch ein Gleichheitszeichen (=) vom Wert getrennt. Der Wert muss mit dem Semikolon (;) abgeschlossen werden. Es entstehen so Paare in der Form <Name>=<Wert>;.

Die Funktion SetValue() setzt die Paare korrekt zusammen. Als erster Parameter wird der Deskriptor des Objektes übergeben, dessen Custom-Eigenschaft gesetzt werden soll. In den Parametern aName und aValue werden der Name und der Wert übergeben. In den alpha-Werten dürfen die Zeichen '=' und ';' nicht enthalten sein.

Um einen Eintrag zu Löschen, wird ein Name mit '' als Wert übergeben.

```
sub SetValue( aObject      : int;  aName       : alpha(20);  aValue      : alpha(8192); ) :
```

Mit der Funktion GetValue() wird die Position des Wertes in der Zeichenkette der Eigenschaft Custom ermittelt und der zugehörige Wert zurückgegeben.

```
sub GetValue( aObject      : int;      // Objektdeskriptor  aName       : alpha(20); // Name
```

Die Funktionen überprüfen nicht die übergebenen Parameter. Dies sollte zumindest in der Funktion SetValue() durchgeführt werden, um zu verhindern, dass die Maximallänge der Eigenschaft Custom überschritten wird und Zeichenketten mit den Zeichen "=" und ";" übergeben werden.

## Kontakt

**Beispiel zum Einbinden einer DLL**

**Aufrufen des Quicksort-Algorithmus aus einer DLL DLL-**

Befehle,  
DLL in

Siehe CONZEPT  
16  
einbinden

In diesem Beispiel wird die Programmierung einer DLL und deren Aufruf in CONZEPT 16 vorgestellt. Um das Beispiel möglichst einfach zu halten, wird ein statisch vordefinierter Array sortiert und zur Ausgabe der externe Debugger verwendet.

Der Quellcode ohne Kommentare befindet sich [hier](#).

Die Implementation besteht aus zwei Teilen: zum einen aus dem Prozedurtext in CONZEPT 16, zum anderen aus dem Quelltext in C. Der Quelltext in C muss in Form einer DLL vorliegen, bevor er in CONZEPT 16 eingebunden werden kann. Bei der Übersetzung der DLL ist darauf zu achten, dass die Funktion C16\_DLLCALL von außerhalb der DLL aufgerufen werden kann. Entsprechende Parameter werden in der Regel beim Linken angegeben.

In den Sourcecode werden die Header-Dateien windows.h und c16\_dll.h eingebunden.

In der c16\_dll.h befinden sich alle notwendigen Definitionen.

```
#include <windows.h>#include <c16_dll.h>
```

Die Funktion QuickSort übernimmt die Sortierung eines Arrays. Ihr wird der vollständige Array übergeben, zusammen mit der Anzahl der zu sortierenden Einträge und dem Bereich der zu sortieren ist.

```
static void QuickSort( char** aTab, // character table vINTs aCount,
```

Zu dem zu sortierenden Bereich wird das mittlere Element ermittelt.

```
{ vINTs LVar; // left variable vINTs RVar; // right variable
```

Die Elemente links und rechts von dem mittleren Element werden solange vertauscht, bis auf der linken Seite alle Elemente, die kleiner und auf der rechten Seite alle Elemente, die größer als das mittlere Element sind.

```
do{ // set left border while (LVar < aCount && strcmp(aTab[LVar],MidElm) < 0) ++LVar; // se
```

Für den vorderen und den hinteren Abschnitt wird noch einmal die Funktion QuickSort aufgerufen.

```
// sort left if (aLoPos < RVar) QuickSort(aTab,aCount,aLoPos,RVar); // sort right if (Iva
```

Die Rekursion wird solange durchgeführt, bis zum Schluss einelementige Abschnitte sortiert sind.

Zu diesem Zeitpunkt liegt eine sortierte Tabelle vor.

Die Funktion QuickSort wird von der Einstiegsfunktion der DLL aufgerufen. Der übergebene Call Control Block enthält die Instanz der DLL und alle notwendigen Funktionsadressen.

## Kontakt

```
vERROR WINAPI C16_DLLCALL( vC16_CCB* aCCB /* call control block */{
```

Über die im Call Control Block übergebene Funktionsadresse können unterschiedliche Funktionen aufgerufen werden. Zunächst wird die Anzahl der übergebenen Argumente mit C16\_ArgCount() ermittelt. Die Funktion wird mit zwei Argumenten aufgerufen: einem Funktionscode und der zu sortierenden Tabelle. In diesem Beispiel wird der Funktionscode nicht benötigt, da nur eine Funktion von der DLL ausgeführt wird. Es zeigt aber, auf welche Weise mehrere Funktionen implementiert werden können. Die Parameter werden beim Prozedur-Befehl DllCall() angegeben.

Wurde mindestens ein Parameter übergeben, werden zu diesem mit der Funktion C16\_ArgInfo() weitere Informationen ermittelt.

```
aCCB->C16_ArgInfo(aCCB->InstHdl, 1, &tType1, NULL, NULL, NULL);
```

Handelt es sich beim ersten Übergabeparameter um einen ganzzahligen Wert, kann die Verarbeitung fortgesetzt werden. Im ersten Parameter steht der Funktionscode. Dieser wird mit der Funktion C16\_ArgRead() gelesen und mit dem anschließenden switch-Konstrukt ausgewertet.

```
if (tType1 == _TypeInt) { /* read command value */ aCCB->C16_ArgRead(aCCB->InstHdl,
```

Soll die Funktion zum Sortieren eines Arrays aufgerufen werden, muss ein weiterer Parameter angegeben werden. Diesem Parameter muss ein Array, bestehend aus Zeichenketten und mit Call-By-Reference, übergeben worden sein.

Für das übergebene Array wird ein Puffer angelegt, bevor der Parameter eingelesen wird. Um die einzelnen Einträge des Arrays zu erreichen, werden die einzelnen Adressen des Arrays in einer Tabelle gespeichert.

```
/* read array */ tBuf = malloc(tLen2); aCCB->C16_ArgRead(
```

Jetzt kann die Funktion QuickSort für die ganze Tabelle aufgerufen werden. Ist die Tabelle sortiert, wird das Ergebnis (die sortierte Tabelle) mit der Funktion C16\_ArgWrite() zurück geschrieben. Da die Tabelle mit der Methode Call-By-Referenz übergeben wurde, steht sie in der CONZEPT 16-Prozedur dann zur Verfügung.

```
QuickSort(tTab, tCount2, 0, tCount2-1); /* store back */ for
```

Jede DLL benötigt eine Funktion, die immer beim Laden und Entladen der Bibliothek aufgerufen wird. Diese Funktion wird vom Betriebssystem gestartet und dient dazu, Funktionsaufrufe der Bibliothek vorzubereiten und Ressourcen zu belegen oder freizugeben.

```
vINT APIENTRY LibMain( HINSTANCE hInstance, DWORD fdwReason, PVOID
```



Die Bibliothek muss so übersetzt werden, dass die Funktion C16\_DLLCALL von außerhalb der DLL aufgerufen werden kann. Dies sind in der Regel Einstellungen, die beim Linken der DLL angegeben werden müssen.

Nachdem die Bibliothek geschrieben und übersetzt ist, kann sie von CONZEPT 16 aus aufgerufen werden. Zunächst werden das Funktionskommando und die maximale

## Kontakt

Größe des Array in einem Define festgelegt.

```
@A+@C+define{ sCmdSort      : 1           // function command  sTabCount      : 16
```

In der Funktion main erfolgt das Füllen des Arrays mit Werten, die Ausgabe des unsortierten Arrays, der Aufruf der DLL und die Ausgabe des sortierten Arrays.

```
mainlocal{ tHdl          : handle;           // DLL handle  tLoop          : int;
```

Die Ausgaben der Funktion erfolgen über den Debugger. Dazu wird zunächst eine Verbindung mit dem Debugger hergestellt. Mit dem Befehl VarAllocate() wird der dynamische Array angelegt und anschließend mit Werten gefüllt.

```
{ DbgConnect('*', false, false); VarAllocate(tStrTab, sTabCount); // define table  tStrTab[ 1]
```

Die Ausgabe des unsortierten Arrays erfolgt in einer Schleife über alle Elemente mit dem Debugger. Der externe Debugger muss zuvor gestartet worden sein.

```
// show table  DbgTrace('Unsortiert:');  for  tLoop # 1;  loop  Inc(tLoop);  while (tLoop <= sTa
```

Mit dem Befehl DllLoad() wird die DLL geladen. Zu diesem Zeitpunkt wird die Funktion LibMain durchgeführt. Da die Einstiegsfunktion den Namen C16\_DLLCALL hat, muss der Name der Einstiegsfunktion nicht als Parameter an DllLoad() übergeben werden.

Der Befehl DllCall() ruft die Einstiegsfunktion mit den angegebenen Parametern auf. Als erster Parameter wird ein Wert übergeben, der die durchzuführende Funktion bestimmt. Der zweite Parameter enthält das zu sortierende Array. Das Array wird mit der Methode Call-By-Reference (var) übergeben, damit der sortierte Array zurückgegeben werden kann.

Mit DllUnload() wird die DLL wieder aus dem Speicher entfernt. Es wird wieder die Funktion LibMain aufgerufen.

```
// sort table  tHdl # DllLoad('DLL\C16TEST');  tHdl->DllCall(sCmdSort, var tStrTab);  tHdl->DllUn
```

Das sortierte Array wird wiederum in einer Schleife über den externen Debugger ausgegeben. Anschließend wird der Speicherbereich für das Array freigegeben und die Verbindung zum Debugger geschlossen.

```
// show table  DbgTrace('');  DbgTrace('Sortiert:');  for  tLoop # 1;  loop  Inc(tLoop);  while
```

In dem hier aufgeführten Beispiel wurden in der DLL nur Funktionen zum Ermitteln der Parameter verwendet. Es stehen über den Call Control Block noch eine ganze Reihe weiterer Funktionen zur Verfügung. Eine vollständige Liste aller Funktionen befindet sich im Abschnitt DLL in CONZEPT 16 einbinden.

## Kontakt

### Beispiel - EvtInit / EvtTerm

Anzeigen eines Dialoges an der zuletzt geöffneten Stelle

Siehe [EvtInit](#),

[EvtTerm](#)

Die Ereignisse [EvtInit](#) und [EvtTerm](#) werden beim Laden und Entladen eines Objektes aufgerufen. In diesem Beispiel wird davon ausgegangen, dass die Funktionen [EvtInit](#) und [EvtTerm](#) bei einem Frame-Objekt bei den entsprechenden Ereignissen eingetragen wurden.

In der Datenbank ist die Datei fPersonalGui angelegt, in der benutzer- und objektabhängig die Koordinaten von Frame-Objekten gespeichert werden.

```
sub EvtInit(  aEvt      : event; // Ereignis) : logic;{  UserInfo(_UserCurrent);  PgiUser # UserI
sub EvtTerm(   aEvt      : event; // Ereignis) : logic;  local  {    tErg : int;  }{  UserInfo(_U
```

## Kontakt

### Beispiel - HLS-Farbwerthe

#### Umrechnung von Farbwerten in das HLS-Modell Farb-

Eigenschaften,  
Farbkonstanten,

Siehe color, Beispiel zu

RGB-Farbwerthen,  
Beispiel ohne  
Beschreibung

In diesem Beispiel wird die Umrechnung von Farbmodellen vorgestellt. Die in CONZEPT 16 verwendeten Farbkonstanten beziehen sich immer auf das RGB-Farbmodell. In diesem Modell wird jede Farbe auf einer Mischung der Farben Rot, Grün und Blau erzeugt. Die jeweiligen Anteile werden durch Farbwerte im Bereich [0..255] angegeben.

Bei Variablen vom Typ color kann über die entsprechenden Eigenschaften der Anteil einer Farbe ermittelt werden. Die Defines mUndefined, mValue2Float und mFloat2Value werden in diesem Text erläutert.

Der vollständige Prozedurtext ohne die erklärenden Kommentare kann hier eingesehen werden.

```
define{ mUndefined : -1.0 mValue2Float(a) : CnvFI(a) / 255.0 mFloat2Value(a) : CnvIF(a *
```

Die meisten Algorithmen zur Umrechnung in andere Farbmodelle basieren auf der Angabe der Farbanteile im Bereich [0..1]. Zur Umrechnung können die Defines mValue2Float (Umrechnung von [0..255] in [0..1]) und mFloat2Value (Umrechnung von [0..1] in [0..255]) verwendet werden.

In diesem Beispiel erfolgt die Umrechnung in das Farbmodell HLS. Dieses Modell besteht ebenfalls aus drei Werten. Der erste Wert bestimmt die Farbe (Hue), der zweite die Helligkeit (Lumination) und der dritte die Sättigung (Saturation). Im Falle das keine Farbe, sondern ein Graustufenwert angegeben werden soll, wird als Farbe der Wert mUndefined angegeben.

Die Umrechnung von RGB-Werten in HLS-Werten erfolgt durch die Funktion Rgb2Hls(). Als Parameter werden zwei Arrays übergeben. Jedes Array besteht aus drei float-Werten. Im ersten Array werden die in [0..1] Werte umgerechneten RGB-Farbanteile übergeben. Als Ergebnis stehen nach der Ausführung der Funktion die HLS-Werte in dem zweiten Array.

```
sub Rgb2Hls( var aRgbColor : float[]; // in: Farbwerte in RGB-Darstellung var aHlsColor : float[]  
// Berechnung der Farbe für Farbdarstellung tDelta # tMax - tMin; switch (true) {
```

Bei der HLS-Darstellung steht in der Komponente Hue ein Winkel, der die Farbe angibt. Hier können also Werte von [0..360] Grad stehen. Die Werte für Helligkeit (L) und Sättigung (S) liegen im Bereich [0..1].

Die Funktion Value() wird zur Umrechnung in RGB-Werte benötigt.

```
sub Value( an1 : float; an2 : float; ahue : float;) : float;{ // Wertebereich überprüfen un
```

## Kontakt

Die Funktion **Hls2Rgb()** berechnet aus den HLS-Werten die entsprechenden RGB-Werte. Als Parameter werden zwei Arrays übergeben. Jedes Array besteht aus drei float-Werten. Im ersten Array werden die HLS-Werte übergeben. Als Ergebnis stehen nach der Ausführung der Funktion die RGB-Werte in dem zweiten Array. Die RGB-Werte müssen noch mit dem Define **mFloat2Value** in den Wertebereich [0..255] transformiert werden, bevor sie mit **ColorRgbMake()** zu einer Farb-Konstanten zusammengesetzt werden können.

```
sub Hls2Rgb( var aHlsColor : float[]; // in: Farbwerte in HLS-Darstellung var aRgbColor : float
```

Mit diesen Funktionen kann auf einfache Weise die Helligkeit von Farben verändert werden:

```
...tColor->vpColorSystem # aEvt:Obj->wpColBkg;tRgbColor[1] # Value2Float(tColor->vpColorR);tRgbCo
```

## Kontakt

**Beispiel - Erstellen und Lesen einer JSON-Datei Anweisungen zum Erzeugen und Lesen einer JSON-Datei Siehe Json-Befehle**

In diesem Beispiel wird eine Funktion zum Erstellen und zum Lesen einer JSON-Datei vorgestellt und Hinweise zur Auswertung der Datei gegeben.

Folgende Funktion erzeugt eine JSON-Datei:

```
sub CreateJson( aFileName : alpha(4096);) local { tDoc : handle; tParent : handle; }
```

Die Funktion erzeugt die übergebene Datei mit festen Werten. Die Datei hat dann folgenden Inhalt:

```
{ "Kreditkarte":"Xema", "Nummer":"1234-5678-9012-3456", "Inhaber":{ "Name":"Reich", "Vor
```

Mit der Anweisung JsonLoad() wird die Datei in einen Baum eingelesen, der anschließend ausgewertet werden kann. Zuvor muss der Wurzelknoten erzeugt werden. Da zur Auswertung sowohl eine unsortierte, als auch eine sortierte Liste verwendet werden soll, werden beide beim Erzeugen angegeben.

```
sub ReadJson( aFileName : alpha(4096);) local { tDoc : handle; tErr : int; }{ tDoc # C
```

Die Auswertung wird durch die Funktion Display vorgenommen. In diesem Beispiel wird in der Funktion ein Dialog geladen, in dem der Inhalt der Datei in einem TreeView-Objekt angezeigt wird. Der Funktion wird der Wurzelknoten des JSON-Dokuments übergeben. Das Füllen des TreeView-Objekts wird durch die Funktion DisplayChildNode() durchgeführt. Die Funktion wird rekursiv aufgerufen.

```
sub Display( aJsonRootNode : handle;) local { tHd1Frame : handle; tNode : handle; }
```

Das hier geladene Fenster-Objekt besteht lediglich aus einem TreeView mit dem Namen tvJson.

```
sub DisplayChildNode( aJsonNode : handle; aTvParent : handle;) local { tJsonNode : handle; case _JsonNodeNumber : { tTvNode->wpNodeStyle # _WinNodeDoc; // convert
```

## Kontakt

### Beispiel - PDF-Dokument drucken

#### Beispiel, wie ein PDF-Dokument erzeugt werden kann

```
@A+@C+main local { tHdlPrtJob : handle; tErr : int; }{ tHdlPrtJob # PrtJobOpen('P
```

In dieser Beispiel-Funktion werden alle Eigenschaften für die PDF-Erzeugung gesetzt.

Notwendig ist lediglich die Eigenschaft PdfFileName. Ohne den Dateinamen wird kein PDF-Dokument erzeugt.

Über die Eigenschaften PdfTitle, PdfAuthor und PdfCreator können die entsprechenden Eigenschaften des PDF-Dokuments besetzt werden.

Mit der Eigenschaft PdfEncryptKeyLen kann die Verschlüsselung bestimmt werden. Die unterschiedlichen Verfahren bestimmen auch, mit welcher Version des Acrobat Readers das Dokument geöffnet werden kann.

Die Passwörter zum Öffnen und zum Ändern der Sicherheitseinstellungen werden in PdfPasswdOpen und PdfPasswdOwner angegeben. Änderungen der Sicherheitseinstellungen sollten immer mit einem Eigentümerpasswort gesichert werden. Die Sicherheitseinstellungen werden in der Eigenschaft PdfRestriction angegeben. Abhängig von der Verschlüsselung des Dokuments (und damit auch der Dokument-Version) stehen mehr oder weniger Rechte zur Auswahl.

Die Qualität des Dokuments bzw. der darin enthaltenen Bilder werden über die Eigenschaften PdfImageResolution, PdfCompression und PdfJpegQuality bestimmt.

## Kontakt

### Beispiel - RGB-Farbwerthe

#### Setzen und Auswerten von RGB-Farbwerthen

Farb-Eigenschaften,

Farbkonstanten,

Siehe color, Beispiel zur

Umrechnung in

andere

Farbsysteme

In den unterschiedlichen Farbeigenschaften können nicht nur die vordefinierten Farbkonstanten, sondern auch beliebige Farben angegeben werden. Diese Farbwerte setzen sich aus den drei RGB-Werten (rot, grün, blau) zusammen.

Farbwerte können einer Variablen vom Typ color zugewiesen werden. Anschließend können die Farbanteile über die Eigenschaften vpColorR, vpColorG und vpColorB ausgelesen werden. Über die Eigenschaften oder die Anweisung ColorRgbMake() können Farbwerte auch zusammengestellt werden.

So kann auch die Farbe von Systemfarben bestimmt werden.

Beispiele:

```
tColor->vpColorSystem # _WinColActiveCaption;tRed    # tColor->vpColorR;tGreen # tColor->vpColorG;
```

## Kontakt

### Beispiel - Schreiben von externen Dateien / Asynchroner Dialog

Erstellen einer externen Datei mit Fsi...-Befehlen und Verwendung asynchroner Dialoge

FsiOpen(),

Siehe FsiWrite(),

WinDialogRun()

Die hier vorgestellte Funktion schreibt den Inhalt einer Datei in eine externe Datei.

Der Funktion wird ein Elternobjekt übergeben. Das Elternobjekt wird verwendet, um den Dialog zur Fortschrittsanzeige an einen bestehenden Frame anzuhängen. Wird in diesem Parameter 0 übergeben, wird kein Elternobjekt verwendet.

Alle weiteren Parameter werden zum Exportieren der Datensätze benötigt. In diesem Beispiel werden die Datensätze einer Artikel-Datei ausgelagert. Es muss die Nummer des Schlüssels und der Pfad und der Name der externen Datei übergeben werden.

Die Parameter, in der das Transparent-Zeichen, der Feld- und der Datensatztrenner angegeben werden, sind optional.

```
sub ArtRecExport( aHdlParent : handle;           // Elternobjekt  aKeyNo      : int;
local{ tHdlDlg       : handle;           // Deskriptor des Dialoges  tHdlProgressBar : handle;
```

Sollten die optionalen Parameter nicht angegeben werden, müssen sie auf Standardwerte gesetzt werden. Anschließend wird die übergebene Datei mit dem Befehl FsiOpen() geöffnet. Die Parameter des Befehls erzeugen die Datei und richten einen exklusiven Schreib-/Lesezugriff ein.

```
// Standardbelegung setzen  if (aMeta  = '')    tMeta   # '~';  else    tMeta   # aMeta;  if (aF
```

Die Datei kann nicht geöffnet werden, wenn sie bereits existiert oder das Verzeichnis nicht vorhanden ist. In diesen Fällen gibt der Befehl einen negativen Wert zurück, der in der weiteren Verarbeitung ausgewertet werden müsste. Um das Beispiel möglichst einfach zu gestalten, wurde auf eine Fehlerbehandlung verzichtet.

Der Dialog, der mit der Anweisung WinOpen() geladen wird besteht aus einem ProgressBar-Objekt und einer Schaltfläche. Die Eigenschaft ProgressPos des Fortschritt-Objekts ist auf 0 gesetzt. Die Schaltfläche ist als "Abbrechen"-Schaltfläche definiert (kann über das Kontextmenü der Schaltfläche im Dialog-Assistenten gesetzt werden).

Der Deskriptor des Progress-Objekts wird noch häufiger benötigt, daher wird er in einer Variablen zwischengespeichert. Es wird dadurch ein besseres Laufzeitverhalten, verglichen mit der Referenzierung über den Namen, erreicht. Die Anzahl der zu exportierenden Datensätze wird in die Eigenschaft ProgressMax eingetragen.

```
// Dialog laden  tHdlDlg # WinOpen('DlgProgress', _WinOpenDialog);  tHdlProgressBar # $ProgressBa
```

Mit dem Befehl WinDialogRun() wird der geladene Dialog gestartet. Wurde ein

Elternobjekt angegeben, wird dieses Objekt mit übergeben. Die Option

WinDialogAsync sorgt dafür, dass nach dem Darstellen des Dialoges die Funktion fortgesetzt wird.

## Kontakt

```
if (aHdlParent != 0)      tHdlDlg->WinDialogRun(_WinDialogAsync, aHdlParent); else      tHdlDlg->Win
```

Nach der Anzeige des Dialoges kann mit dem Auslagern der Datensätze begonnen werden. Dazu wird eine Schleife erstellt, die über alle Datensätze der Artikel-Datei läuft. Die Schleife wird abgebrochen, wenn das Ende der Datei erreicht wird (tErg != \_rNoRec), oder der Benutzer die Abbrechen-Schaltfläche drückt (tHdlDlg->WinDialogResult() != \_WinIdCancel).

Mit dem Befehl WinDialogResult() wird regelmäßig der Zustand des Dialoges überprüft.

Der Befehl gibt die ID der gedrückten Schaltfläche zurück.

```
tErg # RecRead(aFileNo, aKeyNo, _RecFirst); while ((tHdlDlg->WinDialogResult() != _WinIdCancel
```

Nachdem ein Datensatz gelesen wurde, wird die Position des Fortschrittzbalkens um eins erhöht. Mit den Formatierungszeichen und den Feldern aus der Artikel-Datei wird eine Variable zusammengesetzt, die nach der Wandlung in den ANSI-Zeichensatz (StrCnv()) mit dem Befehl FsiWrite() in die externe Datei geschrieben wird.

Anschließend wird der nächste Datensatz gelesen und die Schleife wird erneut durchlaufen.

```
tHdlProgressBar->wpProgressPos # tHdlProgressBar->wpProgressPos + 1; tBuffer # tMeta + Cnv
```

Nachdem der letzte Datensatz gelesen wurde, kann die externe Datei mit dem Befehl FsiClose() geschlossen werden. Anschließend wird mit WinClose() der geladene Dialog geschlossen und die Funktion beendet.

```
tHdlExtFile->FsiClose(); tHdlDlg->WinClose(); return(true);}
```

Die angegebene Funktion exportiert nur die Datensätze aus einer bestimmten Datei. Sie kann aber leicht mit den Befehlen FileInfo(), SbrInfo() und den Fld...()-Befehlen so umgeschrieben werden, dass sie für beliebige Dateien verwendet werden kann.

Um die Funktion möglichst einfach zu halten, findet keine Fehlerbehandlung statt. Fehler können beim Öffnen der externen Datei und beim Laden des Dialoges auftreten. Zudem können fehlerhafte Werte an die Funktion übergeben werden. Diese Prüfungen können auf einfache Weise durch if...else-Abfragen realisiert werden.

Die Funktion verändert den Inhalt der Feldpuffer. Dieses Problem kann beseitigt werden, indem zu Beginn der Inhalt gesichert und am Ende der Funktion wieder hergestellt wird. Das kann mit folgenden Code-Segmenten erfolgen:

```
// Datensatzpuffer sicherntHdlRecBuffer # RecBufCreate(fArtikel); RecBufCopy(fArtikel, tHdlRecBuff
```

Im folgenden die komplette Funktion ohne Anmerkungen:

```
sub ArtRecExport( aHdlParent : handle; // Elternobjekt aKeyNo : int;
{ // Standardbelegung setzen if (aMeta = '') tMeta # '~'; else tMeta # aMeta; if
// Dialog asynchron starten if (aHdlParent != 0) tHdlDlg->WinDialogRun(_WinDialogAsync, aHd
```

## Kontakt

### Beispiel - Selektionen

Selektion öffnen, durchführen und das Ergebnis in einem RecList-Objekt darstellen SelCopy(),

SelOpen(),  
SelRead(),  
Siehe SelClose(),  
SelDelete(),  
DbSelection

In diesem Beispiel wird die Vorgehensweise beim Durchführen einer Selektion vorgestellt. Das nachstehende Beispiel geht von einem Dialog mit einem RecList- und einem Button-Objekt aus.

Beim Drücken der Schaltfläche soll eine Selektion durchgeführt und der Inhalt der Selektionsmenge in der RecList angezeigt werden. Wird die Schaltfläche erneut gedrückt, wird die Selektionsmenge verworfen und der Inhalt der Datei im RecList-Objekt dargestellt.

Die gesamte Verarbeitung kann im Ereignis EvtClicked der Schaltfläche durchgeführt werden.

```
sub EvtClicked( aEvt          : event; // Ereignis ) : logic;local{ tHdlSel : int; tErg
```

Um die Selektion nicht für alle anderen Benutzer der Datenbank zu sperren, wird die Selektion SELA in eine temporäre Selektionsmenge kopiert. Der Name dieser Selektionsmenge setzt sich aus TMP.SELA. und der ID des Benutzers zusammen.

```
tSelName # 'TMP.SELA.' + UserInfo(_UserCurrent);
```

In der folgenden if-Anweisung wird unterschieden, ob im RecList-Objekt bereits eine Selektionsmenge angezeigt wird oder nicht. Um eine Selektionsmenge anzuzeigen, muss die Eigenschaft DbSelection auf den Deskriptor der Selektion gesetzt werden. Beinhaltet die Eigenschaft den Wert 0, wird keine Selektionsmenge angezeigt.

```
if ($RecList->wpDbSelection = 0) {
```

Die vorhandene Selektion SELA wird unter dem temporären Namen kopiert. Auf diese Weise kann die Selektion gesperrt werden, ohne dass andere Benutzer diese Selektion nicht mehr aufrufen können bis sie wieder geschlossen wurde. Da der temporäre Name mit der Benutzer-ID ergänzt ist, kann auch kein anderer Benutzer durch Aufruf der gleichen Prozedur den gleichen Namen ermitteln, da der Benutzer eine andere Benutzer-ID besitzt.

```
tErg # SelCopy(ART.D.Artikel, 'SELA', tSelName);
```

Mit SelOpen() wird ein Selektionspuffer angelegt. Der zurückgegebene Deskriptor wird bei den folgenden Befehlen benötigt. Der Befehl SelRead() liest die kopierte Selektionsmenge und sperrt sie für andere Benutzer. Nur eine gesperrte Selektion kann mit dem Befehl SelRun() durchgeführt werden. Die Optionen SelDisplay und SelBreak sorgen dafür, dass der Lauf der Selektion angezeigt und durch den Benutzer unterbrochen werden kann.

```
tHdlSel # SelOpen();      tErg # SelRead(tHdlSel, ART.D.Artikel, _SelLock, tSelName);      tErg #
```

## Kontakt

Ist die Selektion durchgelaufen, kann der Selektionsdeskriptor der Eigenschaft DbSelection zugewiesen werden. Anschließend findet automatisch ein Neuaufbau der Liste statt, die dann die Datensätze der Selektionsmenge anzeigt.

```
$RecList->wpDbSelection # tHdlSel; }
```

Wird in der Liste bereits der Inhalt einer Selektion angezeigt, wird der else-Fall aufgerufen. Hier wird der Selektionsdeskriptor aus der Eigenschaft des RecList-Objektes ausgelesen und die Selektion mit dem Befehl SelClose() geschlossen. Anschließend wird der Selektionsdeskriptor aus der Eigenschaft gelöscht. Die zuvor kopierte Selektionsmenge wird mit SelDelete() gelöscht.

```
else { // Selektion beenden und löschen SelClose($RecList->wpDbSelection); $RecList->wp
```

Im else-Zweig kann nicht auf die Variable tHdlSel zugegriffen werden, da der Wert der Variable nur im if-Zweig gesetzt wird. Der else-Zweig wird aber erst nach einem erneuten Aufruf der Funktion durchlaufen, die Variable besitzt zu diesem Zeitpunkt keinen gültigen Wert mehr.



Innerhalb der hier vorgestellten Routine werden keine Fehlerüberprüfungen durchgeführt. Dies erfolgt, um das Beispiel möglichst einfach zu halten. In der Regel müssen die Rückgabewerte der einzelnen Befehle entsprechend ausgewertet werden.

Im Anschluss befindet sich noch einmal der gesamte Funktionstext ohne Kommentare:

```
sub EvtClicked( aEvt : event; // Ereignis ) : logic;local{ tHdlSel : int; tErg :
```

## Kontakt

### Beispiel - Selektionszeitpunkt

#### Überprüfung des Alters einer Selektion

Mit der folgenden Funktion kann überprüft werden, ob eine Selektion älter als ein angegebenes Zeitintervall ist. Als Übergabeparameter werden die Datei, in der die Selektion definiert ist, der Selektionsname und ein Zeitintervall in Sekunden angegeben.

```
sub RunSelectionAgain( aFileNo    : int;  aSelName   : alpha(20);  aTimeDiff : bigint; ): logic;
```

In der Funktion wird zunächst die Selektion geöffnet. Hier findet, um das Beispiel nicht zu verkomplizieren, keine Überprüfung statt, ob das gelungen ist.

Anschließend wird der derzeitige Zeitpunkt und der Zeitpunkt der letzten Durchführung der Selektion ermittelt. Beide Zeiten werden in bigint umgerechnet und voneinander abgezogen. Ist die Differenz größer als die in 100 Nanosekunden-Einheiten umgerechnete Zeitdifferenz, liefert die Funktion true, sonst das Ergebnis false zurück.



Bei der Umrechnung von caltime in bigint wird die Eigenschaft vpBiasMinutes berücksichtigt. Die Eigenschaft wird automatisch durch die Methode vmSystemTime() gesetzt. In der anderen Variablen werden nur das Datum und die Uhrzeit gesetzt. Damit die Umrechnung korrekt erfolgt, muss die Zeitzone hier ebenfalls gesetzt werden.

## Kontakt

### Beispiel - Sortierung einer Zeichenkette

Zeichen in einer Zeichenkette alphabetisch sortieren Cte-Befehle,

Siehe Beispiel ohne Beschreibung

In diesem Beispiel wird eine Funktion vorgestellt, die den Inhalt einer Zeichenkette alphabetisch sortiert. Dabei werden alle Buchstaben, die doppelt in der Zeichenkette vorkommen, ignoriert.

Die zu sortierende Zeichenkette wird als erster Parameter übergeben. Im zweiten optionalen Parameter kann bestimmt werden, ob die Groß- und Kleinschreibung berücksichtigt werden soll. Wird in (aOrderCI) true übergeben, werden zunächst die Großbuchstaben und dann die kleinen Buchstaben sortiert.

Als Rückgabewert wird die sortierte Zeichenkette zurückgegeben.

```
sub OrderString( aString      : alpha; // unsortierte Zeichenkette  opt aOrderCI : logic; // Gro
```

Zur Verarbeitung werden einige Variablen benötigt:

```
local{ tHdlTree       : handle;        // Deskriptor des Baums  tHdlItem       : handle;        // D
```

Zuerst wird ein Baum erzeugt, in dem die Buchstaben der Zeichenkette eingepflegt werden. Abhängig davon, ob die Groß- und Kleinschreibung berücksichtigt werden soll, müssen unterschiedliche Bäume angelegt werden.

```
{ // Baum erzeugen  if (!aOrderCI)      tHdlTree # CteOpen(_CteTreeCI);  else      tHdlTree # CteOp
```

Anschließend wird für jedes Zeichen der Zeichenkette ein Element erzeugt und in den Baum integriert. Der Name des Elementes leitet sich dabei aus dem Buchstaben ab.

Existiert zu einem Buchstaben bereits ein Element, kann es nicht in den Baum eingefügt werden und wird gelöscht.

```
// Einlesen der Zeichenkette in einen Baum  for    tCount # 1;   loop  Inc(tCount);  while (tCount
```

Danach wird der Baum sequenziell gelesen und aus den Namen der Elemente die sortierte Zeichenkette zusammengesetzt.

```
// Auslesen des Baums  for    tHdlItem # tHdlTree->CteRead(_CteFirst)  loop  tHdlItem # tHdlTree->
```

Jetzt können alle Elemente aus dem Baum und der Baum selbst gelöscht werden.

```
// Baum leeren und löschen  tHdlTree->CteClear(true);  tHdlTree->CteClose();  return(tStringOrder
```

## Kontakt

Beispiel - switch / EvtMouseItem / EvtLstDataInit

Mehrfachauswahl in einem RecList-Objekt

switch...case...default,

Siehe EvtMouseItem,

EvtLstDataInit

In diesem Beispiel soll die Implementierung einer Mehrfachauswahl in einem RecList-Objekt vorgestellt werden.

Die Datei, deren Datensätze in der RecList angezeigt werden, wurde um ein logisches Feld erweitert. Dieses Feld hat den Wert true, wenn der Datensatz selektiert ist, andernfalls den Wert false. Damit die hier vorgestellten Funktionen auch für RecList-Objekt in anderen Dateien verwendet werden können, wurde der Name des logischen Feldes in der Eigenschaft Custom des RecList-Objektes hinterlegt.

In dem Ereignis EvtMouseItem werden die Mausaktionen verarbeitet. Um einen selektierten Datensatz zu kennzeichnen, wird in der Funktion EvtLstDataInit die Hintergrundfarbe der markierten Datensätze verändert. Die beiden Funktionen müssen den entsprechenden Ereignissen des RecList-Objektes zugewiesen werden.

Im switch-Konstrukt des Ereignisses EvtMouseItem wird nicht wie üblich ein Ausdruck (angegeben hinter dem switch) mit mehreren Konstanten (angegeben hinter dem case) verglichen, sondern das Prinzip umgekehrt. Eine Konstante wird mit mehreren Ausdrücken verglichen. In diesem speziellen Fall können so mehrere Übergabeparameter in einem switch-Konstrukt überprüft werden.

Folgende Fälle werden unterschieden:

- Datensatz mit rechter Maustaste angeklickt

In dem angeklickten Datensatz wird das Markierungsfeld verändert. Ein nicht markierter Satz wird markiert, bei einem markiertem Satz wird die Markierung entfernt.

- Datensatz mit linker Maustaste doppelt angeklickt

Der angeklickte Datensatz wird in einem Dialog angezeigt. Wird der Dialog mit der OK-Taste verlassen, wird der Datensatz zurückgeschrieben. Wird der Dialog abgebrochen, erfolgt keine Speicherung des Datensatzes.



Bei allen Datensatz-Befehlen wurde aus Gründen der Einfachheit, die Überprüfung der Ergebnisse der Datensatzoperationen weggelassen.

In dem Ereignis EvtLstDataInit wird Aufgrund des Markierungsfeldes die Hintergrund Farbe der Spalten gesetzt. Die Spalten werden dabei über eine Schleife ermittelt.

```
sub EvtMouseItem( aEvt      : event;    // Ereignis aButton   : int;      // Maustaste aHitTest : int
                  case ((aButton = _WinMouseLeft | _WinMouseDouble) and (aHitTest = _WinHitLstView)) : {  
sub EvtLstDataInit( aEvt      : event;    // Ereignis aRecID  : bigint;    // Record-ID des
```

## Kontakt

### Beispiel - UserInfo / EvtMenuInitPopup

#### Aktivieren / Deaktivieren von Menüpunkten

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

#### EvtMenuInitPopup

In diesem Beispiel sollen in Abhängigkeit des angemeldeten Benutzers verschiedene Menüeinträge in einem Menü aktiviert bzw. deaktiviert werden. Im Ereignis [EvtMenuInitPopup](#) des Frames wird die unten angegebene Funktion eingetragen. Beim Öffnen eines Menüs oder des Kontextmenüs des Fensters werden dann entsprechende Menüeinträge verändert.

Dazu wird zunächst die Benutzergruppe, die der Benutzer angehört, ermittelt. Gehört der Benutzer keiner Gruppe an, wird der Benutzername als Gruppe verwendet.

Anschließend wird die Eigenschaft Custom überprüft, ob die Benutzergruppe oder der Benutzername in der Eigenschaft eingetragen wurde. Die Eigenschaft Disabled wird dann auf false gesetzt, wenn kein Benutzer oder der aktuelle Benutzer in der Eigenschaft eingetragen ist.

```
sub EvtMenuInitPopup(  aEvt      : event; // Ereignis  aMenuItem : int;    // auslösender Menüpunkt
                      while (tHdlMenuItem != 0) {   switch (true)      {       case (tHdlMenuItem->wpCustom = '') :
```

## Kontakt

### Beispiel - Verknüpfungen

Löschen eines Datensatzes und aller verknüpfter Datensätze RecLink(),

FileInfo(),

Siehe LinkInfo(),

Beispiel ohne  
Beschreibung

In diesem Beispiel werden zwei Funktionen vorgestellt, mit denen sowohl der Datensatz, der sich zur Zeit in den Feldpuffern einer Datei befindet, als auch alle mit diesem Datensatz verknüpften Datensätze gelöscht werden. Durch die Verwendung dieser Funktion kann die "Referenzielle Integrität" der Datenbank gewährleistet werden.

Der vollständige Prozedurcode ohne Kommentare kann hier angezeigt werden.

Die Aufgaben werden auf zwei Funktionen verteilt. Die Funktion DeleteRec() löscht die Datensätze. Diese Funktion wird rekursiv aufgerufen. Die Funktion Delete() ruft die Funktion DeleteRec() auf, nachdem eine Transaktion gestartet wurde. Die Verwaltung der Transaktion findet in der Funktion Delete() statt.

**DeleteRec()**

Dieser Funktion wird die Dateinummer übergeben, in dessen Feldpuffer der zu löschen Datensatz steht. Weitere Überabeparameter werden nicht benötigt.

```
sub DeleteRec( aFileNo : int; // Dateinummer ) : int; local{ tLinkNo      : int; tLinkFileNo : i
```

Zunächst wird der Überabeparameter geprüft. Die Funktion FileInfo(..., \_FileExists) liefert 1 zurück, wenn die angegebene Dateinummer existiert, sonst 0. Sollte die Datei nicht vorhanden sein, wird der Wert ErrFileInvalid zurückgegeben und die Funktion beendet.

```
{ // Überprüfung der Überabeparameter if (FileInfo(aFileNo, _FileExists) = 0) { return(_Er
```

Der Datensatz wird jetzt gelöscht. Können verknüpfte Datensätze nicht gelöscht werden, muss dieser Datensatz wieder hergestellt werden. Dies erfolgt durch die Transaktion.

```
// Datensatz löschen tErg # RecDelete(aFileNo, 0); if (tErg != _rOk) { return(tErg); }
```

In einer Schleife werden alle Verknüpfungen der Datei ermittelt. Existieren keine Verknüpfungen, wird die Zähl-Schleife nicht durchlaufen.

```
// Existiert in dieser Datei eine Verknüpfung? for tLinkNo # 1; loop Inc(tLinkNo); while (t
```

Mit dem Befehl LinkInfo() wird die Zielfile der Verknüpfung ermittelt. Da in dieser Datei Datensätze gelesen werden, erfolgt zunächst eine Speicherung der Feldpuffer dieser Datei. Dazu wird ein neuer Feldpuffer mit dem Befehl RecBufCreate() angelegt und der Inhalt der Feldpuffer mit dem Befehl RecBufCopy() kopiert.

```
// Nummer der Zielfile ermitteln tLinkFileNo # LinkInfo(aFileNo, tLinkNo, _LinkDestFileName
```

## Kontakt

In der folgenden Schleife werden solange Datensätze gelesen, bis entweder kein verknüpfter Datensatz mehr existiert oder ein gesperrter Datensatz gelesen wurde. In diesem Fall bricht die Schleife mit einem entsprechenden Fehler ab, der später ausgewertet wird. Jeder verknüpfte Datensatz, der gelesen werden konnte, wird mit der Funktion DeleteRec() gelöscht. Hier findet ein rekursiver Aufruf statt, da der verknüpfte Datensatz natürlich weitere verknüpfte Datensätze besitzen kann, die ebenfalls gelöscht werden müssen. Um zwischen den Fällen "Fehler beim Löschen des Datensatzes" und "Fehler beim Lesen des Datensatzes" zu unterscheiden, werden die Rückgabewerte beim Löschen um 1000 erhöht.

```
// Verknüpften Datensatz lesen      tErg # RecLink(tLinkFileNo, aFileNo, tLinkNo, _RecLast);      whi
```

Nach dem Durchlauf der Schleife, wird der ursprüngliche Inhalt des Feldpuffers wieder hergestellt. Der angelegte Puffer wird mit dem Befehl RecBufDestroy() wieder entfernt.

```
// Puffer der Zielfile wieder herstellen      tRecBuffer->RecBufCopy(tLinkFileNo);      tRecBuffer->
```

Konnte ein Datensatz nicht gelöscht werden, wird der komplette Aufruf mit einem Fehler abgebrochen. Dies hat zur Folge, dass die komplette Rekursion mit dem gleichen Fehlerwert beendet wird. Der erste Aufruf der Funktion DeleteRec() gibt also ebenfalls den entsprechenden Fehlerwert zurück.

```
// Funktion beenden, wenn ein Datensatz nicht gelöscht werden konnte      if (tErg > 1000) {
```

Sind alle verknüpften Datensätze einer Verknüpfung gelöscht wird im nächsten Schleifendurchlauf der Zähl-Schleife die nächste Verknüpfung behandelt.

Verfügt eine Datei, in der ein Datensatz gelöscht werden soll, über keine Verknüpfung, wird die Zähl-Schleife nicht durchlaufen, d. h. es wird lediglich der Befehl RecDelete() aufgerufen.

Existieren Verknüpfungen, aber keine verknüpften Datensätze, wird die while-Schleife nicht durchlaufen und ebenfalls nur der entsprechende Datensatz gelöscht.

### Delete()

Die Funktion DeleteRec() löscht alle verknüpften Datensätze bis entweder keine verknüpften Datensätze mehr vorhanden sind, oder beim Löschen eines Datensatzes ein Problem (zum Beispiel dieser Datensatz von einem anderen Benutzer bearbeitet wird) auftritt. Um zu verhindern, dass nur ein Teil der verknüpften Datensätze gelöscht werden, wird die Funktion in eine Transaktion eingeschlossen.

Eine Transaktion wird mit dem Befehl DtaBegin() eingeleitet. Verläuft das Löschen der verknüpften Datensätze ohne Probleme, liefert die Funktion DeleteRec() den Wert rOk zurück. In diesem Falle wird die Transaktion mit DtaCommit() beendet und alle Änderungen seit DtaBegin() werden in die Datenbank übernommen. Liefert die Funktion DeleteRec() einen anderen Wert zurück, wird die Transaktion mit DtaRollback() abgebrochen und alle Änderungen seit DtaBegin() werden verworfen.

```
sub Delete( aFileNo : int; // Dateinummer ) : int;local{  tErg : int;}{  DtaBegin();  tErg # Del
```

## Kontakt

Diese Funktionen können in eine beliebige Datenbank eingespielt werden. Durch ihre Programmierung passt sie sich jeder vorhandenen Datenstruktur an. In der Regel wird eine derartige Vielseitigkeit nicht benötigt und die Funktion DeleteRec() kann entsprechend vereinfacht werden.

Diese Funktionen löschen alle verknüpften Datensätze. Einmal generierte Aufträge müssen aber beispielsweise in der Datenbank verbleiben, auch wenn der Kunde dazu gelöscht wurde. In solchen Fällen können bestimmte Verknüpfungen in der Schleife ausgeschlossen werden. Ein anderer Ansatz wäre die Verknüpfungen nicht durch eine Schleife ermitteln zu lassen, sondern sie explizit anzugeben. Dies müsste dann für jede Datei gesondert erfolgen.



Bei der Programmierung mit Transaktionen ist darauf zu achten, dass jedem DtaBegin() ein DtaCommit() oder DtaRollback() folgt. Kommt es während einer Transaktion zu einem Laufzeitfehler, wird die Funktion nicht beendet und damit auch kein DtaCommit() oder DtaRollback() durchgeführt. Die Transaktion ist noch aktiv und wird unter Umständen erst zurückgesetzt, wenn der Benutzer die Datenbank verlässt. Alle Änderungen (auch Änderungen in Prozeduren) werden dann verworfen.

## Kontakt

**Beispiel - Verwendung der COM-Schnittstelle von Microsoft Word Einstieg in die Benutzung der COM-Schnittstelle von Word Siehe Befehle der COM-Schnittstelle**

In diesem Abschnitt wird Anhand von kleinen Beispielen die Benutzung der COM-Schnittstelle von Microsoft Word vorgestellt. Dabei wird nur ein kleiner Ausschnitt der Möglichkeiten vorgestellt.

Eine vollständige Beschreibung der von Microsoft Word zur Verfügung gestellten Objekte, deren Eigenschaften und Methoden, befindet sich im Objektkatalog des Visual Basic-Editors (Menüpunkt "Extras / Makro / Visual Basic-Editor" und anschließend Menüpunkt "Ansicht / Objektkatalog").

### Herstellen einer Verbindung zu Microsoft Word

Die Verbindung zur COM-Schnittstelle von Microsoft Word wird über den Befehl ComOpen() hergestellt. Als Parameter wird der Applikationsname und das Root-Objekt angegeben. Mit der Option ComAppCreate wird ein neuer Word-Prozess gestartet.

```
tComApplication # ComOpen('Word.Application', _ComAppCreate);
```

Der zurückgegebene Deskriptor wird benötigt, um auf untergeordnete Objekte, Eigenschaften und Methoden des COM-Objektes "Application" zugreifen zu können.

### Setzen einer Eigenschaft eines Objektes

Der neu gestartete Prozess wird nicht in der Taskleiste des Betriebssystems angezeigt. Damit die weitere Verarbeitung nachvollzogen werden kann, wird die Eigenschaft "Visible" des Objektes "Application" gesetzt:

```
tComApplication->cplVisible # true;
```

Eine Eigenschaft eines Objektes wird durch die Angabe des Objekt-Deskriptors, der Eigenschaft und des neuen Wertes gesetzt. In dem Beispiel wurde eine Konstante verwendet. Es kann aber auch der Befehl ComPropSet() verwendet werden.

### Aufrufen einer Methode

Eine Methode wird mit dem Befehl ComCall() aufgerufen. Um ein neues Dokument zu erzeugen, muss die Methode "Add" des Objektes "Documents" aufgerufen werden. Das Objekt "Documents" ist dem Objekt "Application" untergeordnet. Die Methode kann auf unterschiedlichen Wegen aufgerufen werden. Ist der Objektpfad ausgehend vom Application-Objekt bekannt, kann dieser in der ComCall()-Anweisung angegeben werden:

```
tComApplication->ComCall('Documents.Add');
```

Die Objekte und die Methode werden durch einen Punkt voneinander getrennt. Wird der Deskriptor zum COM-Objekt "Documents" noch zu einem späteren Zeitpunkt benötigt, kann zunächst das entsprechende Objekt ermittelt und anschließend die Methode aufgerufen werden:

## Kontakt

```
tComDocuments # tComApplication->cphDocuments;tComDocuments->ComCall('Add');
```

### Abfragen von Ergebnissen einer Methode

Liefert eine aufgerufene Methode ein Ergebnis zurück, kann das Ergebnis mit der Anweisung ComCallResult() ermittelt werden.

```
tComApp # ComOpen('Word.Application', _ComAppCreate);tComApp->cplVisible # true; tResult # tComAp
```

### Übertragen von formatierten Daten

Um Text in ein erzeugtes Dokument zu schreiben, muss die Methode "TypeText" des Objektes "Selection" aufgerufen werden. Vor dem Aufruf kann das Format des Textes geändert werden. Zunächst wird das Objekt "Selection" ermittelt. Aus dem Objekt "Selection" wird das Objekt "Font" bestimmt. Über die Eigenschaften des Font-Objektes kann das Format des Textes bestimmt werden.

```
tComSelection # tComApplication->cphSelection;tComFont # tComSelection->cphFont;tComFont->cpaName
```

Jetzt kann mit der Methode "TypeText" der Text übertragen werden:

```
tComSelection->ComCall('TypeText', 'Dear Sir or Madam,' + StrChar(13));
```

### Speichern eines Dokumentes

Die Vorgehensweise zum Speichern eines Dokumentes entspricht der zum Übertragen von Text oder jedem anderen Aufruf einer Methode. Es wird das Objekt ermittelt, das verarbeitet werden soll und dessen Methode wird aufgerufen. Zum Speichern eines Dokumentes muss das Dokument ermittelt und die Methode "Save" oder "SaveAs" aufgerufen werden.

In diesem Beispiel soll das aktive Dokument gespeichert werden. Das Objekt kann über die Eigenschaft "ActiveDocument" des Objektes "Application" ermittelt werden. Der Aufruf zum Speichern des aktiven Dokumentes hat folgendes Aussehen.

```
tComApplication->ComCall('ActiveDocument.SaveAs', _Sys->spPathMyDocuments + '\Com_Example.doc');
```

### Beenden der Verbindung

Wird die Verbindung zwischen CONZEPT 16 und Microsoft Word getrennt, wird Word nicht automatisch beendet. Es muss die Methode "Quit" des Objektes "Application" aufgerufen werden bevor die Verbindung mit ComClose() beendet wird.

```
tComApplication->ComCall('Quit', wdDoNotSaveChanged);tComApplication->ComClose();
```



Das Erstellen von Serienbriefen mit Hilfe der COM-Schnittstelle ist prinzipiell möglich. Durch die Art der Programmierung ist allerdings der Zugriff über die ODBC-Schnittstelle wesentlich flexibler, einfacher zu realisieren und damit auch einfacher zu pflegen. Durch entsprechende Abfragewerkzeuge, die auch von einem ungeübten Benutzer verwendet werden können, kann das Abfragen der Datenbank und das Erstellen von Serienbriefen, Katalogen usw., vollständig durch den Benutzer erfolgen.

## Kontakt

**Web-Schnittstelle - Beispiel einer Web-Anbindung**

**Anwendungsbeispiel der Web-Schnittstelle**

**In einem kleinen Beispiel soll die Kommunikation zwischen dem Browser und der Datenbank verdeutlicht werden. Wesentlich dazu ist die Verbindung von Prozeduren in der Datenbank und den HTML-Seiten.**

**Das Beispiel besteht aus nur einer CONZEPT 16-Prozedur, aus der alle HTML-Seiten generiert werden. Das Programm soll eine vom Benutzer ausgedachte Zahl zwischen 1 und 10 erraten.**

**In der Konfigurationsdatei wurde die Prozedur WebRequest als Request-Funktion eingetragen. Damit wird in der Applikation die main-Funktion der Prozedur aufgerufen, sobald ein Request an die Applikation gestellt wird.**

**Der vollständige Prozedurtext ohne ausführliche Kommentare befindet sich hier.**

**Die Request-Funktion besteht aus einem switch-Konstrukt, das den Aufrufzeitpunkt unterscheidet:**

```
main(){  switch (WseStatus()) {    case _WseUserInit : ...    case _WseUserProc : ...    case _W...
```

**Gibt der Befehl WseStatus() \_WseUserInit zurück, wurde die Request-Funktion zum ersten Mal aufgerufen. Es wird also ein neues Spiel gestartet. Dazu wird ein globaler Datenbereich angelegt, der wie folgt definiert ist:**

```
global Common{  g.fLowest : float; // untere Grenze des Ratebereiches  g.fHighest : float; // ob...
```

**Für den Fall des ersten Aufrufs der Request-Funktion wird folgendes Programmsegment durchlaufen:**

```
case _WseUserInit : // erster Aufruf der Request-Funktion{  VarAllocate(Common); // Seite zurü...
```

**Nachdem die Seite zurückgegeben wurde, kann der Benutzer nur den angegebenen Link betätigen. Dem Link wird das Argument Mode=Starten angefügt. Dieses Argument kann in der Request-Funktion entsprechend ausgewertet werden.**

```
case _WseUserProc : // wiederholter Aufruf der Request-Funktion {  switch (WseArg(_WseArgR...
```

**Die weiteren Argumente müssen ebenfalls ausgewertet werden können. Innerhalb der switch-Konstruktion switch (WseArg(\_WseArgReq, 'Mode'))**

**müssen also noch die Fälle GuessLow, GuessRight und GuessHigh abgefragt werden.>**

**Die Fälle GuessLow und GuessHigh unterscheiden sich nur in der Berechnungsvorschrift des nächsten Ratevorgangs. Je nach Fall wird die untere Grenze des Ratebereiches nach oben oder die obere Grenze nach unten verschoben.**

```
case 'GuessLow' : // zu niedrig geraten {      Inc(g.iTry); // unteren Bereich...
```

**Im Falle das richtig geraten wurde, wird die Anzahl der Versuche ausgegeben und der Benutzer abgemeldet.**

## Kontakt

```
case 'GuessRight' : // richtig geraten { // Anzahl der Versuche anzeigen
```

Vor dem Entfernen des Web-Benutzers wird die Request-Funktion nochmals aufgerufen. In diesem Fall wird von WseStatus() der Wert WseUserTerm zurückgegeben. Hier wird lediglich der in WseUserInit angelegt Datenbereich freigegeben.

```
case _WseUserTerm : // letzter Aufruf der Request-Funktion { VarFree(Common); } // End
```

## Kontakt

### Modern Theme Style

#### Moderne Themes

Über Modern Theme Style lässt sich die Darstellung der graphischen Benutzerschnittstelle anpassen. Farben, Farbverläufe, Schriften und Metriken von Oberflächen-Objekten lassen sich individuell konfigurieren, so dass ein Design nach eigenen Vorstellungen erstellt werden kann. Das System bietet darüber hinaus auch eine einfache Möglichkeit, die Darstellung an das Erscheinungsbild von Windows 10 anzupassen.

### Themes

In der bisherigen Verfahrensweise wird die Darstellung von Oberflächen-Objekten über Eigenschaften gelöst. Die Schriftfarbe einer Schaltfläche etwa, kann über die Eigenschaft ColFg beeinflusst werden. Wenn eine einheitliche Farbgebung erreicht werden soll, müssen die Eigenschaften aller betroffenen Objekte entsprechend verändert werden. Ein weiterer Nachteil ist, dass für viele Objekte keine Darstellung erreicht werden kann, die unter allen Betriebssystemen identisch ist. Schaltflächen unter Windows 7 haben z. B. eine anderes Aussehen als unter Windows 10.

Modern Theme Style hingegen arbeitet auf der Basis von Themes. Ein Theme stellt das Design für die Darstellung der graphischen Benutzeroberfläche aller unterstützten Oberflächenobjekte bereit. Um die Darstellung einer Anwendung zu ändern reicht somit die Aktivierung des gewünschten Themes.

### Theme-Objekt

Ein Theme wird unter CONZEPT 16 durch das Theme-Objekt repräsentiert. Hierbei handelt es sich um ein Storage-Objekt, dass analog zu Dialogen, PrintForms, etc. anhand eines Namens in der Datenbank gespeichert und von dort wieder geladen werden kann.

CONZEPT 16 stellt darüber hinaus vorgefertigte Themes bereit:

- \_OfficeBlue - Theme, das sich am Aussehen von Microsoft Office 2016 (blaue Farbgebung) orientiert.
- \_OfficeDark - Theme, das sich am Aussehen von Microsoft Office 2016 (dunkle Farbgebung) orientiert.
- \_WindowsColor - Theme, das die Systemfarben von Windows verwendet. Dies ist insbesondere dann von Vorteil, wenn unter Windows die Einstellung "Hoher Kontrast" in der Systemsteuerung aktiviert ist. Hier können Anwender mit eingeschränkter Sehfähigkeit, die Farbgebung unter Windows auf ihre Bedürfnisse einstellen.



Vordefinierte Themes beginnen immer mit einem Unterstrich (\_) vor dem eigentlichen Name und werden nicht in der Datenbank abgelegt.

#### Erstellung eines Theme

Ein Theme kann sowohl im Theme-Editor als auch prozedural über den Befehl WinThemeOpen() erstellt werden und besteht aus einer Liste von Theme-Elementen. Die Liste umfasst alle über Modern Theme Style beeinflussbaren Oberflächenobjekte. Über entsprechende Eigenschaften lässt sich das Design eines Objekttyps anpassen.

## Kontakt

### Beispiel

```
sub CreateMyTheme()  local  {      tTheme : handle;  }{  // Theme erstellen.  tTheme # WinThemeOpen
```

Das Beispiel erstellt ausgehend vom vordefinierten Theme \_OfficeBlue ein neues Theme. Hierzu wird der Befehl WinThemeOpen() aufgerufen. Der ermittelte Deskriptor bezeichnet das neu erstellte Theme-Objekt. Anschließend werden einige Farben für Button-Objekte (\_WinTypeButton) gesetzt. Zum Schluss wird das Theme unter dem Name MyBlueTheme in der Datenbank gespeichert.

### Unterstützte Objekte

Die folgende Liste zeigt alle Objekte, die aktuell Modern Theme Style unterstützen und im Theme-Editor bzw. prozedural angepasst werden können.

- Animation
- AppFrame
- Axis
- BigIntEdit
- Button
- Checkbox
- CodeEdit
- ColorButton
- ColorEdit
- ColumnCheckbox
- ColumnRadiobutton
- DataList
- DataListPopup
- DataListPopupColumn
- DateEdit
- DecimalEdit
- Divider
- DocView
- Edit
- FloatEdit
- FontNameEdit
- FontSizeEdit
- Frame
- GanttGraph
- GroupBox
- GroupSplit
- GroupTile
- HyperLink
- Icon
- IntEdit
- Label
- MdiFrame
- MenuButton
- MenuItem
- MetaPicture
- Notebook

## Kontakt

- NotebookPage
- Picture
- PopupColor
- PopupList
- Progress
- PrtPreviewDlg
- Radiobutton
- RecList
- RecListPopup
- RecListPopupColumn
- RtfEdit
- Scrollbox
- Slider
- Statusbar
- Statusbar-Button
- StoList
- StoListPopup
- StoListPopupColumn
- TextEdit
- TimeEdit
- Toolbar
- Toolbar-Button
- ToolbarDock
- ToolbarMenu
- TreeNode
- TreeView
- WebNavigator
- Windowbar

### Theme-Aktivierung

Aktiviert wird der Modern Theme Style, indem die Eigenschaft StyleTheme auf WinStyleThemeModern gesetzt wird. Diese Einstellung bezieht alle Unterobjekte, deren Eigenschaft StyleTheme auf WinStyleThemeParent gesetzt ist, mit ein. Der Name für das zu verwendende Theme wird über die Eigenschaft ThemeName spezifiziert. Ist diese Eigenschaft beim Objekt nicht vorhanden, wird die Eigenschaft ThemeName des App-Objektes herangezogen.

### Anwendungsobjekt

Die Eigenschaft ThemeName ist im App-Objekt enthalten und wirkt sich auf folgende Objekte aus:

- Objekte ohne ThemeName-Eigenschaft

Diese Objekte verwenden das in App eingetragene Theme, wenn die Eigenschaft StyleTheme des Objektes auf WinStyleThemeModern gesetzt ist.

- Objekte mit ThemeName-Eigenschaft

Diese Objekte verwenden das in App eingetragene Theme, wenn die Eigenschaft StyleTheme des Objektes auf WinStyleThemeModern gesetzt ist

## Kontakt

und die Eigenschaft ThemeName beim Objekt leer ist.



Die Eigenschaft StyleTheme kann beim App-Objekt nicht auf WinStyleThemeModern gesetzt werden.

### Frame-Aktivierung

Zur Aktivierung des Modern Theme Style für einen bestimmten Frame wird der Deskriptor des gewünschten Frame-Objektes verwendet.

```
$Frame->wpThemeName # 'MyBlueTheme'; $Frame->wpStyleTheme # _WinStyleThemeModern;
```

### Aktivierung für Container-Objekte

Container-Objekte (wie GroupBox, NotebookPage, etc.) verfügen über die Eigenschaft ThemeName, so dass pro Container ein separates Theme aktiviert werden kann.

### Ausschluss untergeordneter Objekte

Standardmäßig wirkt sich das Setzen der Eigenschaft StyleTheme auch auf untergeordnete Objekte des angegebenen Oberflächenobjektes aus, da StyleTheme bei allen Objekten auf WinStyleThemeParent initialisiert ist. Um einzelne Objekte auszunehmen genügt es den Wert der Eigenschaft entsprechend zu setzen.

```
$Button->wpStyleTheme # _WinStyleThemeSystem;
```

Hierdurch wird die systemabhängige Darstellung für die Schaltfläche aktiviert (und somit Modern Theme Style deaktiviert).

### Layer-Modus

Die Aktivierung eines Theme kann je nach Umfang der betroffenen Objekte dazu führen, dass ein sichtbarer Aufbau der Objekte am Bildschirm erfolgt. Durch den Befehl WinLayer() kann das verhindert werden.

```
// Während des Theme-Wechsels, Hinweis für 1 Sekunde aktivieren.WinLayer(_WinLayerStart, $Frame,
```

### Menü-Darstellung bei aktiviertem Modern Theme Style

Wenn Modern Theme Style für ein Frame-Objekt aktiviert ist, kann systembedingt kein Hauptmenü (Eigenschaft MenuName) angezeigt werden. Als Ersatz bietet CONZEPT 16 das ToolbarMenu-Objekt an. Hierbei handelt es sich um ein spezielles Toolbar-Objekt, welches vollautomatisch das über MenuName gesetzte Menü emuliert.

### Theme-Mixing / ThemeSets

Unter Umständen ist es notwendig ein Theme für einzelne Unterobjekte zu modifizieren, jedoch nicht für alle Objekte des gleichen Typs. Um in diesen Fällen nicht das Theme kopieren zu müssen, besteht die Möglichkeit ThemeSets mittels WinThemeSetOpen() oder des Theme-Editors anzulegen. Das ThemeSet enthält die gleichen Theme-Elemente und Eigenschaften, wie ein Theme, erlaubt jedoch

## Kontakt

abweichende Werte für einzelne Eigenschaften einzelner Objekte. Eigenschaften, die im ThemeSet nicht definiert sind, erben den Wert, den diese Eigenschaft im Theme besitzt. Ein Theme kann beliebig viele ThemeSets enthalten. Das jeweilige ThemeSet des Objekts wird über die Eigenschaft ThemeSetId in Verbindung mit den Eigenschaften StyleTheme und ThemeName ausgewählt.

## Kontakt

### Externer Debugger

#### Beschreibung des externen Debuggers

Siehe [Blog](#)

Der CONZEPT 16-Debugger ist ein externes Programm, das zur Überwachung von Prozeduren und Funktionen verwendet wird. Im Gegensatz zum integrierten Debugger können damit auch Prozeduren und Funktionen getestet werden, die mit RmtCall() oder C16\_ProcCall() (externe Programmierschnittstelle) aufgerufen wurden oder von der Web-Schnittstelle verwendet werden.

Nach dem Starten des Debuggers erscheint ein Applikationsfenster. Ohne eine Verbindung zu einem Client-Programm kann der Debugger nur wieder geschlossen werden. Eine Verbindung zum Debugger kann mit dem Prozedurbefehl DbgConnect() oder aus dem Editor des CONZEPT 16-Clients über den Menüpunkt Prozedur / Debugger verbinden hergestellt werden. Zur Kommunikation wird der TCP/IP-Port 4721 verwendet.

Für jede Debuggerverbindung wird ein MDI-Fenster (das Verbindungsfenster) dargestellt. Die Verbindung kann drei Zustände annehmen: Verbindung hergestellt, Verarbeitung angehalten und Verbindung getrennt. Der Zustand der Verbindung wird in der Systemschaltfläche des MDI-Fensters angezeigt. In Abhängigkeit vom Zustand stehen unterschiedliche Funktionen zur Verfügung.

Bei einer bestehenden Verbindung kann mit dem Debugger die Verarbeitung von Funktionen unterbrochen werden. Ist die Verarbeitung angehalten, können Feld- und Variableninhalte überprüft und die Funktion im Einzelschritt-Modus durchlaufen werden. Ist der Bereich, der untersucht werden sollte durchlaufen, kann die Verarbeitung normal fortgesetzt werden.

Soll die Verarbeitung bei bestimmten Anweisungen angehalten werden, können sowohl im Debugger, als auch im Editor des Designers Breakpoints gesetzt werden. Der Debugger hält die Verarbeitung an, bevor die Zeile in der der Breakpoint gesetzt wurde, durchgeführt wird.



Das Überwachen der Prozeduren sollte nicht auf dem Computer erfolgen, auf dem die Prozeduren verarbeitet werden.

Der Debugger kommuniziert über das TCP/IP-Protokoll mit dem entsprechenden Client. Damit können beispielsweise Prozeduren auf dem Web-Server von einer Arbeitsstation aus getestet werden. Die Beschreibung des Debuggers gliedert sich in folgende Bereiche:

- Installation des Debuggers
- Starten des Debuggers
- Das Debugger-Fenster

## Kontakt

### Installation des Debuggers

#### Beschreibung der Installation des CONZEPT 16-Debuggers

Die Programmdateien des externen Debuggers befinden sich im Programmstand des CONZEPT 16-Clients. Es müssen keine weiteren Anpassungen vorgenommen werden.

Folgende Dateien werden dabei für den Debugger benötigt:

#### *Dateien des Debuggers*

c16_chart_w32.dll	Chart-Bibliothek (32-Bit)
c16_chart_w64.dll	Chart-Bibliothek (64-Bit)
c16_coded_w32.dll	CodeEdit-Bibliothek (32-Bit)
c16_coded_w64.dll	CodeEdit-Bibliothek (64-Bit)
c16_debg.dll	Debug Message Bibliothek
c16_debg.exe	Debugger (32-Bit)
c16_debg_w64.exe	Debugger (64-Bit)
c16_diff_w32.dll	Difference-Bibliothek (32-Bit)
c16_diff_w64.dll	Difference-Bibliothek (64-Bit)
c16_graph_w32.dll	Graphics-Bibliothek (32-Bit)
c16_graph_w64.dll	Graphics-Bibliothek (64-Bit)
c16_icu_w32.dll	Unicode-Bibliothek (32-Bit)
c16_icu_w64.dll	Unicode-Bibliothek (64-Bit)
c16_obj_w32.dll	GUI-Objekte (32-Bit)
c16_obj_w64.dll	GUI-Objekte (64-Bit)
c16_pdf_w32.dll	PDF-Bibliothek (32-Bit)
c16_pdf_w64.dll	PDF-Bibliothek (64-Bit)
c16_res_w32.dll	GUI-Ressourcen (32-Bit)
c16_res_w64.dll	GUI-Ressourcen (64-Bit)
c16_ssl_w32.dll	SSL-Bibliothek (32-Bit)
c16_ssl_w64.dll	SSL-Bibliothek (64-Bit)
c16_sys_w32.dll	System-Objekte (32-Bit)
c16_sys_w64.dll	System-Objekte (64-Bit)
c16_xml_w32.dll	XML-Bibliothek (32-Bit)
c16_xml_w64.dll	XML-Bibliothek (64-Bit)
c16_zip_w32.dll	ZIP-Bibliothek (32-Bit)
c16_zip_w64.dll	ZIP-Bibliothek (64-Bit)



Falls sich der Debugger nicht auf dem gleichen Rechner wie der verwendete CONZEPT 16-Client befindet, muss der TCP/IP-Port 4721 des Debugger-Systems für eingehende Verbindungen geöffnet sein.

## Kontakt

### Starten des Debuggers

#### Debugger starten

Der Debugger wird mit dem Programm C16\_DEBG.EXE gestartet. Beim Starten des Debuggers sind folgende Möglichkeiten zu unterscheiden:

- Starten über eine Verknüpfung auf dem Desktop

Ob der Debugger minimiert oder maximiert gestartet wird, hängt von den Einstellungen in der Verknüpfung ab und überschreibt in diesem Falle die in der Konfigurationsdatei des Debuggers gespeicherte Einstellung.

Ist in der Verknüpfung die Einstellung "Normal" angegeben, werden die Einstellungen aus der Konfigurationsdatei bezogen.

- Starten über den Editor

Wird im Editor der Menüpunkt "Debugger verbinden" gewählt, wird der lokale Debugger immer minimiert gestartet, unabhängig von den Einstellungen in der Konfigurationsdatei des Debuggers.

- Starten durch manuellen Aufruf der Datei c16\_debg.exe

Es wird die übergebene Konfigurationsdatei zur Restauration aller gespeicherter Einstellungen verwendet, bzw. die Datei c16\_debg.ini, wenn keine Konfigurationsdatei in der Kommandozeile angegeben wurde.

Solange kein Client mit dem Debugger eine Verbindung hergestellt hat, können lediglich die Einstellungen und die Informationen über den Debugger aufgerufen werden. Der Debugger legt die Datei C16\_DEBG.INI zur Speicherung der Einstellungen an.

Die zu verwendende Konfigurationsdatei kann in einem Übergabeparameter beim Starten des Debuggers angegeben werden:

c16\_debg.exe [-ini=<dateiname>]

Ist die Option "ini" nicht angegeben oder <dateiname> leer, wird die c16\_debg.ini im Konfigurationsverzeichnis des Clients (z. B. C:\ProgramData\CONZEPT 16\Client) geladen bzw. erzeugt, falls dort nicht vorhanden.

Ansonsten handelt es sich bei <dateiname> um die komplette Pfadangabe inklusive Dateiname. Ist eine Dateiendung angegeben, wird diese ignoriert. Die Dateiendung ist immer .ini.

Existiert die angegebene Datei nicht, wird beim Start des Debuggers eine Warnung ausgegeben. Anschliessend startet der Debugger mit den Standardeinstellungen. Beim Verändern der Konfiguration im Konfigurationsdialog oder Beenden des Debuggers werden die aktuellen Einstellungen in der angegebenen Konfigurationsdatei gesichert.

Welche Konfigurationsdatei gerade vom Debugger verwendet wird, kann über die Titelleiste des Konfigurations-Dialgues nachgeprüft werden. Dort wird der Name der aktuell verwendeten Konfigurationsdatei angezeigt.

## **Kontakt**

**Beispiele:**

- Konfigurationsdatei "user1.ini" im Arbeitsverzeichnis laden bzw. anlegen.

**c16\_dbg.exe -ini=user1**

- Konfigurationsdatei "user1.ini" im Verzeichnis "c:\users" laden bzw. anlegen.

**c16\_dbg.exe -ini=c:\users\user1**

**Auf diese Weise kann für jeden Benutzer eine individuelle Konfigurationsdatei angelegt werden.**

**Der Debugger kann mit dem Menüpunkt "Datei / Beenden" wieder geschlossen werden.**

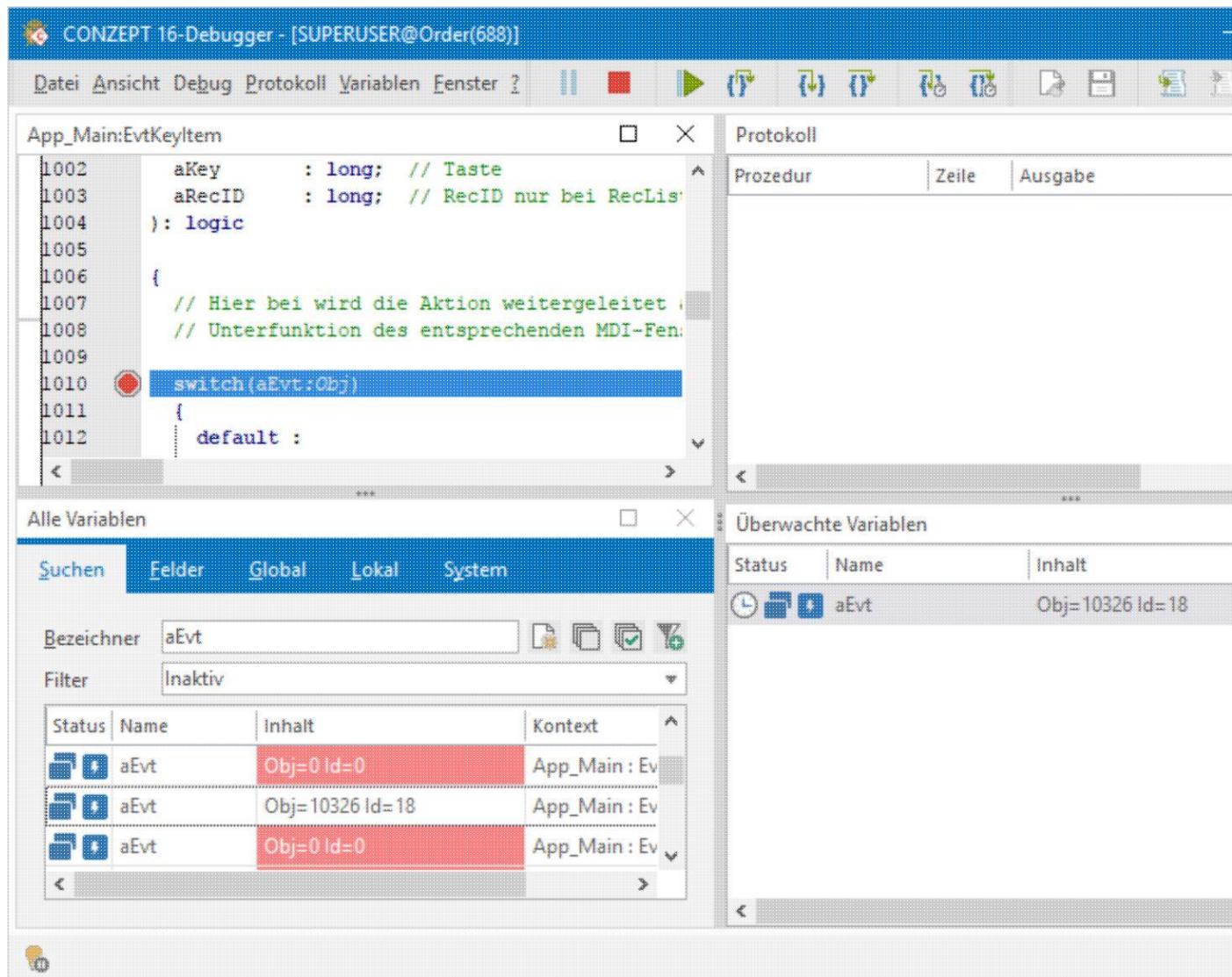
## Kontakt

### Das Debugger-Fenster

#### Der Aufbau des Debugger-Fensters

Der Debugger ist als MDI-Anwendung konzipiert. Jede Verbindung mit dem Debugger wird in einem MDI-Fenster angezeigt und kann getrennt von den anderen Verbindungen bearbeitet werden.

Nach dem Starten des Debuggers und dem Herstellen einer Verbindung erscheint folgendes Fenster:



Das Fenster des externen Debuggers teilt sich in folgende Bereiche:

- **Menüzeile**

Der Debugger wird über die Menüeinträge gesteuert. Alle Funktionen mit Ausnahme der Funktionen "Einstellungen" und "Beenden" betreffen nur die Verbindung, deren MDI-Fenster gerade aktiv ist.

- **Werkzeugeleisten**

## **Kontakt**

**Verschiedene Funktionen des Menüs lassen sich ebenfalls über die Werkzeugleiste aufrufen.**

- **Verbindungsfenster**

**Das Verbindungsfenster teilt sich in verschiedene Bereiche zur Darstellung des Prozedurtextes, des Laufzeitprotokolls, der Variablen und der überwachten Variablen auf. Die einzelnen Bereiche können in der Größe verändert oder ein- bzw. ausgeblendet werden.**

## Kontakt

### Das Debugger-Menü

#### Erläuterungen zur Menü-Struktur des Debuggers

Die Menüstruktur gliedert sich in folgende Menüs:

- Datei
- Ansicht
- Debug
- Protokoll
- Variablen
- Fenster
- Hilfe (?)

Funktionen, die ebenfalls über die Werkzeugeiste angesprochen werden können, sind mit dem entsprechenden Symbol in der Werkzeugeiste gekennzeichnet.

#### Datei

In diesem Menü stehen folgende Funktionen zur Verfügung:

- Sitzung beenden 

Mit dieser Funktion wird die Verbindung zwischen dem Debugger und der Anwendung getrennt. Ob eine Sicherheitsabfrage erfolgt, kann in den Einstellungen mit dem Punkt Abfrage beim Beenden einer Sitzung bzw. Abfrage beim Speichern definiert werden.

- Alle Sitzungen beenden

Mit dieser Funktion werden alle Verbindungen zwischen dem Debugger und den Anwendungen getrennt. Ob eine Sicherheitsabfrage erfolgt, kann in den Einstellungen mit dem Punkt Abfrage beim Beenden einer Sitzung bzw. Abfrage beim Speichern definiert werden.

- Einstellungen ...

Die Einstellungen des Debuggers werden aufgerufen. Die Einstellungen betreffen alle Verbindungen. Folgende Einstellungen können vorgenommen werden:

Im Bereich Allgemein können generelle Einstellungen (zum Beispiel zur Anzeige und zu Abfragen) vorgenommen werden.

- ◆ Anzeigesprache

In der Auswahlliste lassen sich die Sprachen auswählen, die zur Anzeige der Debuggeroberfläche verwendet werden können.

- ◆ Name des Themes

Hier kann ein Standard-Theme ausgewählt werden, in dem der Designer dargestellt wird. Folgende Einstellungen sind hier möglich:

## Kontakt

à Office (Blau) – Blaues Theme, welches an Office 2016 angelehnt ist.

à Office (Dunkel) – Dunkles Theme, welches an Office 2016 angelehnt ist.

### “ Symbole passend zum Theme einfärben

Ist diese Einstellung gesetzt, werden die Symbole im Designer in der zum Theme passenden monochromen Farbe eingefärbt, andernfalls werden mehrfarbige Symbole verwendet.

### “ Abfrage beim Beenden der Sitzung

Wird das Verbindungsfenster geschlossen erfolgt eine Trennung der Verbindung zum Client. Ist diese Option gesetzt, erfolgt vor der Trennung eine Sicherheitsabfrage.

### “ Abfrage beim Löschen

Ist diese Option gesetzt, wird beim Löschen von Variablen zuvor eine Sicherheitsabfrage vorgenommen.

### “ Abfrage beim Speichern

Ist diese Option aktiviert, wird beim Beenden des Debuggers oder beim Laden einer Variablenliste nachgefragt, ob die Liste der Variablen gespeichert werden soll. Die Abfrage erscheint nur, wenn die Liste der Variablen nach dem letzten Speichern verändert wurde.

### “ Schneller Einfügemodus beim Suchen

Mit dieser Option kann bestimmt werden, ob eine Variable oder ein Feld, das im Prozedurtext mit einem Doppelklick ausgewählt wurde, im Bereich "Alle Variablen" auf der Suchen-Seite oder im Bereich der "Überwachten Variablen" eingefügt wird. Ist die Option aktiv, wird die Variable sofort zu den überwachten Variablen hinzugefügt.

Hier werden Einstellungen vorgenommen, welche die Durchführung von Prozeduren im Debugger beeinflussen.

### “ Zeitverzögerung bei Automatik

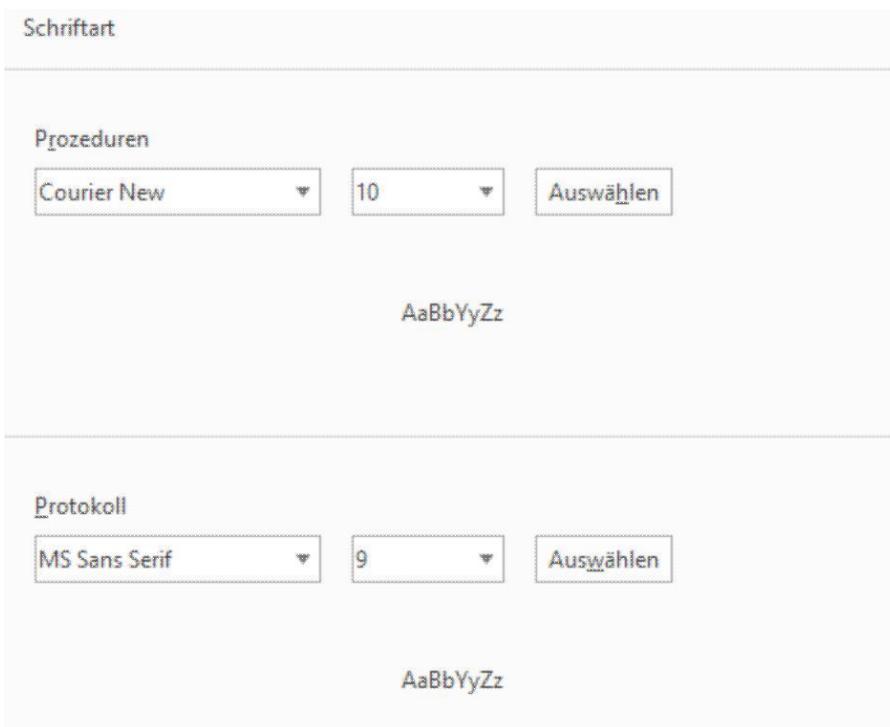
Wird die Funktion Einzelschritt Automatik oder Prozedurschritt Automatik aufgerufen, werden in einem hier vorgegebenen Zeitintervall die Anweisungen durchgeführt. Das Zeitintervall wird in Millisekunden angegeben.

..

Breakpoint setzen mit  -Taste

In der aktuellen Zeile der Prozedur kann mit der Taste  ein Breakpoint gesetzt werden. Durch Aktivierung der Option wird die Tastenkombination  +  zum Setzen eines Breakpoints verwendet.

## Kontakt



In dem Bereich "Schriftart" können die Zeichensätze für den Debugger angegeben werden. Der Zeichensatz zur Darstellung der Prozeduren und für das Protokoll können unterschiedlich eingestellt werden.

Änderungen des Fonts können über die Combobox oder über die Schaltfläche [Auswählen] erfolgen. Der durch die Schaltfläche geöffnete Dialog ermöglicht neben der Änderung des Fonts noch Einstellungen des Schriftschnittes und des Schriftgrades. Der Schriftgrad kann im Bereich 1 bis 38 frei gewählt werden.

Mit der Schaltfläche [Standard ...] wird die Schriftart Courier New in 10 pt und MS Sans Serif in 9 pt für Prozeduren und das Protokoll eingetragen.

### ♦ Farben

Hier werden die Farben für die einzelnen Syntaxtypen im Debugger eingestellt. Durch Anklicken eines Wertes kann eine neue Farbe aus einer ComboBox ausgewählt werden.

### ♦ Benutzerdefinierte Farben

Ist ein Hacken bei *Benutzerdefinierte Farben* gesetzt, werden die Farben aus der Liste verwendet. Andernfalls werden die Farben des definierten Themes (siehe Einstellungen) benutzt.

### ♦ Schaltfläche [Standard]

Über diese Schaltfläche wird das Standard Farb-Schema eingestellt. Im unteren Bereich befindet sich eine Vorschau der aktuell eingestellten Farben.

## Kontakt

Hier werden Einstellungen vorgenommen, welche die Anzeige im Debugger beeinflussen.

- ◆ Exponential-Darstellung für Float-Werte

float-Werte werden in der Variablenliste in Exponential-Darstellung angezeigt.

- ◆ Variabldetails anzeigen

In der Variablenliste werden Details der ausgewählten Variable angezeigt.

Die Einstellungen werden erst mit Drücken der Schaltfläche [OK] gespeichert. Geänderte Einstellungen wirken sich sofort auf alle geöffneten Verbindungen aus. Alle Einstellungen werden in der Datei c16\_debg.ini gespeichert (siehe [Speicherorte von Konfigurationsdateien](#)).

- Beenden

Bestehen noch Debugger-Verbindungen, erfolgt eine Sicherheitsabfrage, bevor der Debugger beendet wird.

## Ansicht

Über diesen Menüpunkt werden die im [Verbindungsfenster](#) angezeigten Informationen ausgewählt. Die Einträge Prozedur, Protokoll, Alle Variablen und Überwachte Variablen aktivieren und deaktivieren die entsprechenden Bereiche.

Die "Einzelansicht" zeigt jeden der Bereiche für sich fensterfüllend an.

Über den Menüpunkt "Schema oben / unten" wird gesteuert, welcher der Verschiebebalken im Fenster über die gesamte Breite bzw. Höhe geht. Ist das horizontale Schema aktiv, geht der horizontale Trenner über die gesamte Breite des Fensters. Wird ein Anzeigebereich ausgeblendet, wird der Anzeigebereich daneben über die gesamte Breite dargestellt.

Ist das vertikale Schema aktiv, wird der vertikale Trenner über die gesamte Höhe des Fensters angezeigt. Wird ein Bereich ausgeblendet, wird der darüber- oder der darunterliegende Bereich vergrößert.

Die zur Verfügung stehenden Werkzeugeleisten und die Statuszeile kann über die Menüpunkte Symbolleisten... und Statuszeile aktiviert und deaktiviert werden.

## Debug

Die Funktionen in diesem Menü beeinflussen die Verarbeitung der Prozeduren. Die Funktionen stehen nur bei einer vorhandenen Verbindung zur Verfügung. Folgende Funktionen können aufgerufen werden:

- Prozedur anhalten 

Die Verarbeitung der Prozedur wird angehalten, sobald ein Funktionsaufruf durch das System erfolgt. Aufrufe durch das System erfolgen beim Ausführen

## Kontakt

einer Prozedur und wenn ein Ereignis ausgelöst wurde. Durch erneutes Aufrufen des Menüpunktes wird beim Aufruf eines Ereignisses oder einer Prozedur nicht mehr gestoppt.

- Prozedur abbrechen 

Die Verarbeitung wird abgebrochen. Es werden nicht nur die laufende Funktion, sondern auch alle aufrufenden Prozeduren oder Funktionen abgebrochen.

- Prozedur fortsetzen 

Die angehaltene Funktion wird fortgesetzt. Die Verarbeitung wird erst wieder bei dem Befehl DbgStop() oder beim nächsten Breakpoint unterbrochen.

- Fortsetzen nur akt. Funktion 

Die aktuelle Funktion wird bis zum Ende fortgesetzt. Die Verarbeitung wird erst unterbrochen, wenn der Befehl DbgStop() ausgeführt, die Verarbeitung durch einen Breakpoint unterbrochen oder in die aufrufende Funktion zurückgekehrt wird. Die Verarbeitung stoppt nur dann nach dem Rücksprung in die aufrufende Funktion, wenn durch die Verarbeitung keine externen Aufrufe (zum Beispiel Ereignisse) ausgelöst wurden.

- Einzelschritt 

Mit dieser Funktion wird der nächste Befehl ausgeführt. Handelt es sich dabei um einen Funktionsaufruf, wird in die Funktion verzweigt und die Verarbeitung vor dem ersten Befehl der Funktion angehalten.

- Prozedurschritt 

Mit dieser Funktion wird der nächste Befehl ausgeführt. Bei einem Funktionsaufruf wird die Funktion komplett verarbeitet und erst nach der Rückkehr des Funktionsaufrufs die Verarbeitung angehalten.

- Einzelschritt Automatik 

Die Verarbeitung wird fortgesetzt, allerdings werden die einzelnen Anweisungen in bestimmten Zeitabständen ausgeführt. Das Intervall kann in den Einstellungen definiert werden. Standardmäßig werden zwei Befehle pro Sekunde ausgeführt. Bei Funktionsaufrufen wird in die entsprechende Funktion verzweigt.

- Prozedurschritt Automatik 

Die Verarbeitung wird fortgesetzt, allerdings werden die einzelnen Anweisungen in bestimmten Zeitabständen ausgeführt. Das Intervall kann in den Einstellungen definiert werden. Standardmäßig werden zwei Befehle pro Sekunde ausgeführt. Bei Funktionsaufrufen wird die Funktion in normaler Geschwindigkeit durchgeführt und nach der Rückkehr verlangsamt fortgesetzt.

- Breakpoints

In diesem Untermenü können Breakpoints gesetzt und entfernt werden. Ist im Anzeigebereich der Prozedur eine Zeile markiert, kann diese Zeile mit einem Breakpoint versehen werden. Dazu wird der Menüpunkt "Breakpoint ein / aus" angewählt. Über den gleichen Menüpunkt kann der Breakpoint wieder entfernt werden.

## Kontakt

Über das Kontextmenü im linken Rand des Prozedurbereiches können für beliebige Zeilen Breakpoints gesetzt werden. Die anderen Funktionen entsprechen den Funktionen im Menü.

Mit den Funktionen "Prozedur-Breakpoints löschen" und "Alle Breakpoints löschen" werden alle Breakpoints in der Prozedur bzw. in allen Prozeduren entfernt.

## Protokoll

Im Protokollbereich werden alle Zeichenketten, die dem Befehl DbgTrace() übergeben wurden, ausgegeben. Darüber hinaus können alle Prozedur- und Funktionsaufrufe hier angezeigt werden. Mit den Anweisungen DbgControl( DbgTimeStart) und DbgControl( DbgTime) können die Laufzeiten von Funktionen und Funktionsabschnitten ausgegeben werden.

- Auswahl kopieren 

Ausgewählte Zeilen im Protokollbereich werden in die Zwischenablage kopiert.

- Auswahl löschen

Ausgewählte Zeilen im Protokollbereich werden entfernt.

- Leeren 

Der Inhalt des Protokollbereichs wird geleert.

- Speichern 

Der Inhalt des Protokollbereichs wird in eine externe Datei gespeichert. Die Datei kann über einen beliebigen ASCII-Editor (zum Beispiel Notepad) gelesen werden.

- Aufrufe extern 

Alle Funktionsaufrufe vom System (zum Beispiel Ereignisse) werden protokolliert.

- Aufrufe intern 

Alle Funktionsaufrufe werden protokolliert. Dies ist nur möglich, wenn die Prozeduraufrufe protokolliert werden.

- Funktionsaustritt 

Der Rücksprung der Funktionsaufrufe wird protokolliert. Dies ist nur möglich, wenn die "Aufrufe extern" protokolliert werden.

## Variablen

Im Verbindungsfenster können die Inhalte von Felder und Variablen eingesehen werden. Über Funktionen in diesem Bereich können neue Variablen zu den überwachten Variablen hinzugefügt oder entfernt werden. Zudem kann die Liste der Variablen zur späteren Verwendung gespeichert werden.

Folgende Funktionen können aufgerufen werden:

## Kontakt

- Neu... 

Befinden sich im Bereich der überwachten Variablen Felder oder Variablen, wird mit dieser Funktion die bestehende Liste geleert und es können neue Variablen und Felder eingefügt werden. Wurde die Liste gespeichert, wird ebenfalls der Name der Liste entfernt.

- Öffnen... 

Wurde eine Liste mit Variablen gespeichert, kann sie mit dieser Funktion wieder geladen werden. Werden bereits Variablen angezeigt, werden diese durch die Liste ersetzt. Der Name der Variablen-Liste wird auf den Namen der geladenen Liste gesetzt.

- Speichern 

Mit dieser Funktion wird die Liste der Variablen unter dem angezeigten Namen gespeichert. Ist noch kein Name angegeben, wird die Funktion "Speichern unter ..." aufgerufen.

- Speichern unter...

Die Liste der angezeigten Variablen wird unter einem Namen gespeichert. Der Name wird in dem aufgerufenen Dialog angegeben.

- Leeren 

Mit dieser Funktion wird die Liste der angezeigten Variablen geleert. Der Name der Liste wird dabei nicht verändert.

- Sortieren nach ...

Die Variablen können nach unterschiedlichen Kriterien sortiert werden. Die entsprechende Sortierung kann in dem untergeordneten Menü ausgewählt werden:

- ◆ Gruppe - Die Variablen werden nach ihrer Zugehörigkeit sortiert. Zuerst werden die Felder, anschließend die globalen und die lokalen Variablen und zum Schluss die Systemvariablen angezeigt.
- ◆ Datentyp - Die Variablen werden nach ihren Typen sortiert (alpha, int, float ...).
- ◆ Name - Die Variablen werden alphabetisch nach dem Namen sortiert.
- ◆ Kontext - Die Variablen werden alphabetisch nach dem Kontext sortiert.

- Variable löschen 

Die in der Liste ausgewählte Variable wird aus der Liste entfernt.

- Variable überwachen 

Für die in der Liste ausgewählte Variable wird die Überwachung aktiviert. Die Verarbeitung wird angehalten, sobald eine Änderung des Inhaltes erfolgt.

- Exponential-Darstellung für Float-Werte

float-Werte werden in der Variablenliste in Exponential-Darstellung angezeigt.

## Fenster

## Kontakt

In diesem Menü können verschiedene Verbindungsfenster im Debugger angeordnet oder ausgewählt werden:

- Anordnen

Alle Verbindungsfenster werden hintereinander kaskadierend angeordnet.

- Symbole anordnen

Alle zu Symbolen verkleinerten Verbindungsfenster werden neu im Debugger angeordnet.

- Übereinander

Alle geöffneten Verbindungsfenster werden übereinander angezeigt.

- Nebeneinander

Alle geöffneten Verbindungsfenster werden nebeneinander angezeigt.

- Nächstes / Vorhergehendes

Mit dieser Funktion kann zwischen den Verbindungsfenstern gewechselt werden.

- Fensterliste

In dem untergeordneten Menü sind alle Verbindungen aufgelistet. Durch die Auswahl einer Verbindung, wird diese in den Vordergrund geholt.

## Hilfe (?)

- Hilfe

Über diesen Menüpunkt wird die CONZEPT 16-Online-Dokumentation aufgerufen.

- Info

Hier finden sich allgemeine Informationen über den CONZEPT 16-Debugger, wie beispielsweise die Versionsnummer.

- ♦ Bibliothek

In diesem Abschnitt werden die verwendeten Bibliotheken mit ihren Versionen angezeigt. Dies umfasst sowohl die CONZEPT 16-Bibliotheken, als auch die vom Betriebssystem zur Verfügung gestellten Bibliotheken.

- ♦ Parameter

Unter dem Registerreiter "Parameter" werden Informationen zum Speicherort der verwendeten Konfigurationsdatei, sowie zu Kommandozeilenargumenten angezeigt:

Konfiguration Speicherort der verwendeten  
Debugger-Konfigurationsdatei  
INI

## **Kontakt**

**Kommandozeilenargument zum Auffinden der  
Konfigurationsdatei**

◆ **System**

Hier befinden sich die gleichen Informationen, wie beim Info-Dialog des Designers  
(siehe Menü Hilfe (?)).

## Kontakt

### Die Debugger-Werkzeugeleisten

#### Verwendung der Werkzeugeleisten im Debugger

Im Debugger werden vier verschiedene Werkzeugeleisten angezeigt. Jede der Leisten kann durch Ziehen am Ziehpunkt aus der Position im Debugger-Fenster entfernt und an eine beliebige Stelle auf dem Bildschirm platziert werden. Die Reihenfolge und Position der Werkzeugeleisten am oberen Fensterrand kann ebenfalls beliebig verändert werden. Die Position der Werkzeugeleisten wird beim Beenden des Debuggers gespeichert und beim erneuten Starten wieder hergestellt.

Über die Schaltflächen der Werkzeugeleiste werden Funktionen aufgerufen, die ebenfalls über das Menü zur Verfügung stehen. Folgende Funktionen können in den entsprechenden Werkzeugeleisten aufgerufen werden:

#### Datei

In dieser Werkzeugeleiste befinden sich Funktionen aus dem Datei-Menü:

-  Datei / Sitzung beenden ... Die Verbindung zwischen Debugger und Anwendung wird getrennt.
-  Datei / Beenden Der Debugger wird beendet.

#### Debug

In dieser Werkzeugeleiste befinden sich Funktionen aus dem Menü Debug.

-  Debug / Prozedur anhalten Die Verarbeitung der Prozedur wird angehalten.
-  Debug / Prozedur abbrechen Die Prozedur und alle aufrufenden Prozeduren werden abgebrochen.
-  Debug / Prozedur fortsetzen Die Verarbeitung der Prozedur wird fortgesetzt.
-  Debug / Fortsetzen nur akt. Funktion Die Prozedur wird fortgesetzt. Am Ende der Prozedur wird die Verarbeitung angehalten.
-  Debug / Einzelschritt Der nächste Befehl wird ausgeführt. In Funktionsaufrufe wird verzweigt.
-  Debug / Prozedurschritt Der nächste Befehl wird ausgeführt. Funktionsaufrufe werden vollständig durchgeführt.
-  Debug / Einzelschritt Automatik Verarbeitung mit zwei Befehle pro Sekunde fortsetzen. In Funktionsaufrufe wird verzweigt.
-  Debug / Prozedurschritt Automatik Verarbeitung mit zwei Befehle pro Sekunde fortsetzen. Funktionsaufrufe werden in normaler Geschwindigkeit durchgeführt.

#### Protokoll

In dieser Werkzeugeleiste werden Funktionen des Menüs Protokoll aufgeführt.

-  Protokoll / Leeren Der Protokollbereich wird geleert.
-  Protokoll / Speichern Der Inhalt des Protokollbereiches wird in einer Datei gespeichert.
-  Protokoll / Aufrufe extern Prozeduraufrufe vom System protokollieren.
-  Protokoll / Aufrufe intern Funktionsaufrufe protokollieren.
-  Protokoll / Funktionsaustritt Funktionsrücksprung protokollieren

## Kontakt

### Variablen

In dieser Werkzeugleiste befinden sich Funktionen aus dem Menü Variablen.

-  Variablen / Neu Neue Variablenliste anlegen.
-  Variablen / Öffnen Gespeicherte Variablenliste einlesen.
-  Variablen / Speichern Variablenliste speichern.
-  Variablen / Leeren Angezeigte Variablen entfernen.
-  Variablen / Variable löschen Eine Variable aus der angezeigten Liste entfernen.
-  Variablen / Variable überwachen Eine angezeigte Variable auf Änderungen überwachen.

### Breakpoints

In dieser Werkzeugleiste befinden sich Funktionen aus dem Menü Debug / Breakpoints.

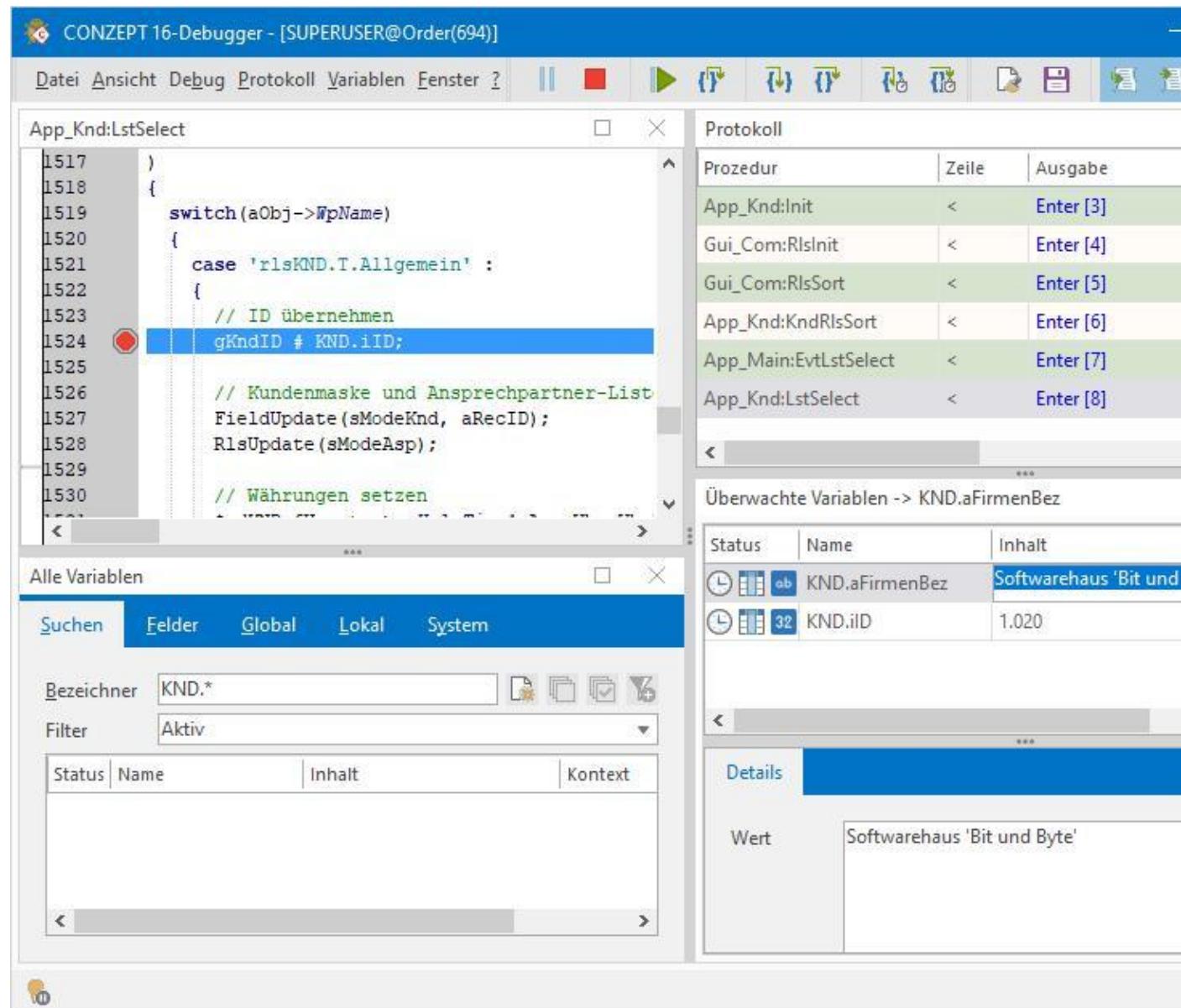
- Debug / Breakpoints / Breakpoint ein / aus Breakpoint in der aktuellen Zeile ein- oder ausschalten.
- Debug / Breakpoints / Alle Breakpoints löschen Alle Breakpoints löschen.

## Kontakt

### Das Debugger-Verbindungsfenster

#### Beschreibung des Verbindungsfensters

Für jede Verbindung zwischen einer Anwendung und dem Debugger wird ein eigenes Verbindungsfenster angezeigt.



In der Titelzeile des Verbindungsfensters und in der Statusleiste des Debuggers wird der Zustand der Verbindung angezeigt:

- 💡 - Es besteht keine Verbindung zu einer Applikation.
- 💻 - Es besteht eine Verbindung zu einer Applikation auf einem anderen Rechner oder mit einer Version älter als 5.8.04.
- 💡 - Es besteht eine Verbindung zu einer Applikation auf dem gleichen Rechner.
- ⌚ - Die Verarbeitung einer Prozedur wurde angehalten.



## Kontakt

Der Zustand in der Titelzeile wird nur bei nicht maximierten Verbindungsfenstern angezeigt. Der Zustand in der Statusleiste bezieht sich immer auf das aktuelle Verbindungsfenster.

Der Debugger stellt eine Verbindung zum **CONZEPT 16-Server** her und meldet sich dort mit einem speziellen Benutzer an die Datenbank an. Kann der Benutzer nicht angemeldet werden (zum Beispiel, wenn in die Datenbank zu diesem Zeitpunkt ein Update eingelesen wird, die Datenbank ist dann gesperrt), gibt der Debugger eine entsprechende Fehlermeldung aus.



Der Debugger kann nur dann verwendet werden, wenn in der Datenbank der Prozedurtext vorhanden ist. Ohne Prozedurtext ist eine Fehlersuche nicht möglich.

Das Verbindungsfenster ist in vier Bereiche aufgeteilt:

- Prozedur
- Protokoll
- Alle Variablen
- Überwachte Variablen

Die einzelnen Bereiche können über das Menü "Ansicht" ein- und ausgeblendet werden. Über den Eintrag "Einzelansicht" wird der gesamte Fensterbereich einem der Bereiche zur Verfügung gestellt.

In einigen Bereichen werden Listen dargestellt. In allen Listen kann die Breite der Spalten durch Ziehen der Spaltenzwischenräume zwischen den Spaltenüberschriften verändert werden. Durch Anklicken der Spaltenzwischenräume kann die ursprüngliche Spaltenbreite wieder hergestellt werden. Die Reihenfolge der Spalten kann durch Verschieben der Spaltenköpfe geändert werden.

### Prozedur

Ist die Verarbeitung einer Prozedur angehalten, wird die Prozedur in diesem Bereich dargestellt. Die Zeile mit der Anweisung, die als nächstes ausgeführt wird, ist mit einem farbigen Balken markiert.

Innerhalb der Prozedur kann mit einem Doppelklick auf eine Variable oder ein Feld die Variable in die Liste der überwachten Variablen aufgenommen werden. Existieren mehrere Variablen mit dem gleichen Namen, werden alle Variablen in die Liste der gefundenen Variablen eingetragen. Aus dieser Liste kann dann die Variable aus dem gewünschten Kontext in die Liste der zu überwachenden Variablen übernommen

werden. Durch Anklicken der Variable bei gleichzeitigem Drücken der -Taste wird die Variable des aktuellen Kontextes direkt in die Liste der zu überwachenden Variablen aufgenommen.

Wird mit der Maus über ein Feld oder eine Variable in der aktiven Funktion gefahren, erscheint ein Tooltip mit dem Inhalt. Dieser wird je nach Datentyp in mehreren unterschiedlichen Interpretations- und Darstellungsformen angezeigt. Hier am Beispiel für den Datentyp int:

## Kontakt

KND.iID: Int	
Wert:	1.020 (0x000003fc)
Datum:	1902-10-16
Zeit:	00:00:01,02
Farbe (RGB):	252,3,0
Farbe (RGB-Hex):	fc,03,00

Wird im linken Rand das Kontextmenü aufgerufen, können innerhalb der Prozedur Breakpoints gesetzt werden. Die Verarbeitung der Prozedur kann dann fortgesetzt werden und wird erst wieder beim nächsten Breakpoint unterbrochen.

Mit folgenden Tastenkombinationen können Suchleisten in der Prozeduransicht angezeigt werden:

-  +  Aktiviert die Suchleiste mit der Funktion "Suchen".
-  +  +  Aktiviert die Suchleiste mit der Funktion "Gehe zu Funktion".
-  +  Aktiviert die Suchleiste mit der Funktion "Gehe zu Zeile".

## Protokoll

Im Protokollbereich werden alle Ausgaben, die mit dem Befehl DbgTrace() durchgeführt werden, eingetragen. Darüber hinaus kann über das Protokoll-Menü der Aufruf und der Rücksprung von bzw. aus Prozedur und Funktionen protokolliert werden.

Mit den Anweisungen DbgControl( DbgTimeStart) und DbgControl( DbgTime) kann die Laufzeit von Funktionen oder Teilbereichen von Funktionen gemessen werden. Die Ausgabe der Laufzeiten erfolgt im Protokollbereich.

Der Inhalt des Protokolls kann mit den entsprechenden Funktionen in einer externen Datei gespeichert werden. Einträge können auch per Mehrfachauswahl in die Zwischenablage kopiert werden.

## Alle Variablen

In diesem Bereich kann über die entsprechende Notizbuchseite auf verschiedene Felder und Variablen zugegriffen werden. Der Zugriff ist nur möglich, wenn die Verarbeitung einer Funktion unterbrochen wurde.

### • Suche

Auf dieser Seite kann eine Suche über alle Felder, globale Variablen und lokale Variablen durchgeführt werden. Bei der Suche können die Wildcards \* und ?

angegeben werden. Nach Drücken der Taste  werden alle gefundenen Variablen, Felder und Argumente in der darunterliegenden Liste dargestellt. Einzelne Variablen können durch Doppelklick in die Liste der überwachten Variablen übernommen werden. Zusätzlich können mehrere Variablen markiert

werden und mit der Taste  überwacht werden. Über die Schaltflächen  und  können keine bzw. alle Einträge in der Liste selektiert werden.

## Kontakt

Die Schaltfläche trägt alle gefundenen Variablen in die überwachten Variablen ein. Mit Drücken der Schaltfläche kann die Liste geleert werden.

Bei der Suche werden alle Felder und alle bereits angelegten Variablenbereiche berücksichtigt. Dies sind die lokalen Variablen der aufgerufenen Funktionen und die globalen Variablen, deren Bereiche in den aufgerufenen Funktionen allokiert wurden. Dabei spielt es keine Rolle, ob eine lokale Variable noch gültig ist oder der globale Variablenbereich mittlerweile wieder freigegeben wurde.

Ist zum Zeitpunkt der Suche eine Datenbank mit der Anweisung DbaConnect() verbunden, werden die Felder dieser Datenbank ebenfalls durchsucht.

Über die Auswahlliste im Punkt Filter lassen sich die zu suchenden Datentypen und Datenbereiche einschränken. Dabei werden die markierten Einträge aus beiden Kategorien mit UND und innerhalb einer Kategorie mit ODER kombiniert.

In der Liste werden die gefundenen Felder und Variablen angezeigt. Neben dem Namen der Variablen oder des Feldes werden der Kontext (Funktion, in der die Variable gültig ist bzw. Datei-, Teildatensatz- und Feldnummer) und der Inhalt angezeigt.

Wird durch die Suche nur ein Feld oder eine Variable gefunden, wird das Feld bzw. die Variable sofort in die Liste der überwachten Variablen übernommen, wenn in den Einstellungen die Option Schneller Einfügemodus beim Suchen aktiviert ist.



Je nach Größe des Bereiches Alle Variablen werden nicht alle Objekte auf der Suchseite angezeigt.

- Felder

Auf dieser Seite können alle Felder der Datenstruktur eingesehen werden.

- Global

Hier werden alle globalen Variablenbereiche, die in den aufgerufenen Funktionen angelegt wurden, angezeigt. Die Variablenbereiche werden auch dann angezeigt, wenn sie wieder freigegeben wurden.

- Lokal

Auf dieser Seite werden die lokalen Variablenbereiche der bislang aufgerufenen Prozeduren angezeigt. Der Fokus steht auf dem Bereich der Funktion, die zurzeit durchgeführt wird.

- System

Im diesem Bereich werden Systemvariablen und die aktiven Funktionen angezeigt. Der globale Fehlerwert und der aktuelle Speicherverbrauch (siehe ProcessMemoryKb) können mit einem Doppelklick in die Liste der überwachten Variablen aufgenommen werden. Ein Anhalten bei Änderungen in den Systemvariablen ist allerdings nicht möglich.

## Kontakt

Die aktiven Funktionen sind alle Funktionen, in die bei einer weiteren Verarbeitung wieder zurückgesprungen wird. Es sind also die Funktionen, die Unterfunktionen aufgerufen haben, die noch nicht beendet wurden.

Um den Wert einer Variablen oder eines Feldes zu überprüfen, muss auf der entsprechenden Seite der Eintrag doppelt angeklickt werden. Der Name wird dann in die Liste der überwachten Variablen aufgenommen.

Auf den Seiten Felder, Global und Lokal kann gesucht werden. Wird in dem jeweiligen Baum die Taste oder die Tastenkombination + gedrückt, kann in einem Eingabefeld ein Name aus dem Baum angegeben werden. Nach Abschließen der Eingabe mit wird auf das entsprechende Element positioniert. Die Verwendung von Wildcards (\*, ?) ist bei der Suche möglich. Sind keine Wildcards angegeben wird nach allen Stellen gesucht, die den Suchbegriff enthalten (\*<Begriff>\*).

## Überwachte Variablen

In diesem Bereich können die Inhalte von Feldern, globalen und lokalen Variablen und Systemvariablen eingesehen werden. Um eine Variable oder ein Feld in die Liste aufzunehmen, kann entweder in der dargestellten Prozedur oder im Alle Variablen-Bereich der Name der Variablen oder des Feldes doppelt angeklickt werden. Erzielt die Suche nur einen Treffer, wird diese Variable oder dieses Feld automatisch in die Liste der überwachten Variablen aufgenommen, wenn in den Einstellungen die Option Schneller Einfügemodus beim Suchen aktiviert ist. In die Liste der überwachten Variablen können bis zu 1000 Variablen eingetragen werden.

Von überwachten Feldern oder Variablen wird der Status, der Name, der Wert und der Kontext angezeigt.

In der Spalte "Status" werden Informationen zu dem Überwachungsstatus und der Variablen angezeigt:

- Überwachungsstatus:

Der Debugger kann die Werte von verschiedenen Variablen und Feldern kontrollieren. Tritt in diesen Variablen oder Feldern eine Veränderung des Wertes auf, unterbricht der Debugger automatisch die weitere Verarbeitung der Funktion. Die Verarbeitung bleibt nach der Anweisung, die die Änderung hervorgerufen hat, stehen. Wird die Veränderung durch eine Benutzerinteraktion verursacht (zum Beispiel Selektieren eines Datensatzes in einem RecList-Objekt), bleibt die Verarbeitung an der nächsten möglichen Stelle stehen (zum Beispiel beim Aufruf des nächsten Ereignisses).

- ◆ grüne Markierung - Verarbeitung der Prozedur wird angehalten, sobald sich der Wert der Variablen ändert.
- ◆ rote Markierung - Die Verarbeitung der Prozedur wurde angehalten, weil sich der Wert der Variablen geändert hat.
- ◆ graue Markierung - Die Variable wird nicht kontrolliert.

Die Systemvariablen können nicht kontrolliert werden.

## Kontakt

Die Markierung kann zur Laufzeit der Prozedur über einen Doppelklick geändert werden.

- Herkunft:

Die Herkunft der Variablen wird über Buchstaben gekennzeichnet:

- ◆ F - Datenbankfeld
- ◆ G - globale Variable
- ◆ L - lokale Variable
- ◆ S - Systemvariable

- Feld- oder Variablen-Typ:

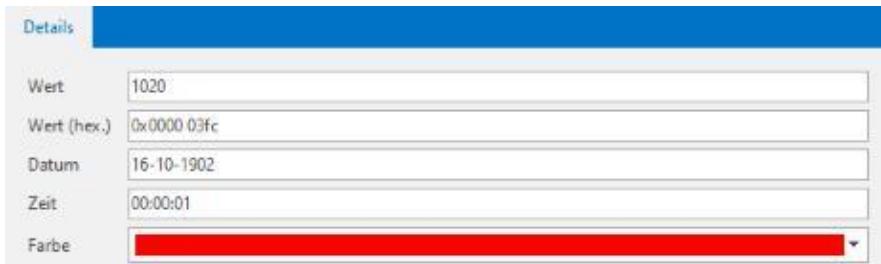
In der ersten Spalte wird ebenfalls der Typ der Variablen über ein entsprechendes Symbol angezeigt.

Je nach verwendeter Sortierung steht in der Spaltenüberschrift noch eine "1" oder "2". Werden die Variablen nach dem Status sortiert, können sie entweder in der Reihenfolge ihrer Herkunft (1) oder ihres Typs (2) sortiert sein. Durch nochmaliges Anklicken der Spaltenüberschrift kann zwischen den beiden Sortierungen gewechselt werden.

Im Inhalt wird der Wert der entsprechenden Variablen oder des Feldes dargestellt. Der Inhalt kann durch Doppelklick in die Spalte "Inhalt" oder durch Drücken der

-Taste verändert werden. Durch erneutes Drücken der Eingabetaste wird der Wert übernommen. Der Inhalt von Variablen vom Typ event und caltime, sowie Variablen außerhalb ihres Gültigkeitsbereiches können nicht editiert werden. Wird eine Variable oder ein Feld bearbeitet, steht der Name im Titel des Bereiches in der Form -> Name.

Zusätzlich werden die unterschiedlichen Interpretationsmöglichkeiten des jeweiligen Datentyps für Variablen und Felder mit mehreren Möglichkeiten unterhalb der Liste dargestellt. Diese können in den angezeigten Feldern auch bearbeitet werden. Hier am Beispiel für Datentyp int:



The screenshot shows a 'Details' tab in a software application. It contains five input fields: 'Wert' (Value) with the value '1020', 'Wert (hex.)' with the value '0x0000.03fc', 'Datum' (Date) with the value '16-10-1902', 'Zeit' (Time) with the value '00:00:01', and 'Farbe' (Color) with a red color swatch. Below these fields is a dropdown menu.

Die Anzeige der Variabldetails kann in den Einstellungen unter Anzeige ein- bzw. ausgeblendet werden.

Bei Variablen, die zurzeit keine Gültigkeit haben (weil zum Beispiel die Funktion, in der die Variablen definiert ist, nicht aufgerufen ist), wird der Inhalt der Variablen rot hinterlegt.

## Kontakt

Bei den Datentypen **point** und **rect** wird der Cursor in die entsprechende Komponente positioniert. Anschließend kann der Wert über die Schaltflächen an der rechten Seite des Eingabefeldes oder durch die Eingabe eines neuen Wertes verändert werden.

Wird in den Wert des Datentyps **font** geklickt, wird ein Eingabefenster angezeigt:



In diesem Fenster wird die Schriftart und der Schriftgrad angezeigt. Durch Anklicken in den betreffenden Bereich kann ein anderer Font oder Schriftgrad gewählt werden. Die Schaltflächen unterhalb des Eingabefeldes zeigen die Attribute der Schrift an. In der Reihenfolge der Schaltflächen sind das Fett, Kursiv, Unterstrichen und Durchgestrichen. Durch das Verlassen des Feldes wird die Variable entsprechend gesetzt.

Im Kontext wird bei Feldern die Dateinummer, Teildatensatznummer und die Feldnummer angegeben. Bei lokalen Variablen ist hier die Funktion angegeben, in der sie deklariert ist. Bei globalen Variablen wird der Name des Variablenbereiches angezeigt.

Die Liste kann nach dem Namen der Variablen oder Typ durch Anklicken der entsprechenden Spaltenüberschriften sortiert werden. Die Sortierung nach Typ wird dabei zunächst nach der Herkunft der Variablen (Felder, lokale Variablen, globale Variablen) vorgenommen. Durch erneutes Anklicken der Spaltenüberschrift erfolgt die Sortierung nach dem Typ (**alpha**, **int**, ...) der Variablen.

Eine Variable oder ein Feld kann mit der -Taste aus der Liste entfernt werden. Dazu muss zuvor der Eingabefokus auf den betreffenden Eintrag gestellt werden.

## Kontakt

### CONZEPT 16-Server

#### Beschreibung des CONZEPT 16-Servers

Performance  
unter

Siehe

VMware



Eine Auflistung der Betriebssysteme, auf denen der Server betrieben werden kann, finden Sie in den Systemvoraussetzungen.

Ein CONZEPT 16-Datenbanksystem besteht immer aus mindestens einem CONZEPT 16-Datenbankserver und den CONZEPT 16-Clients wie beispielsweise dem CONZEPT 16 Standard-Client.

Der CONZEPT 16-Server ist hierbei für das Verwalten aller Datenbanken zuständig. Während die einzelnen Clients ihre Anfragen ausschließlich an den Server richten, bearbeitet dieser alle eingehenden Anfragen und liefert die Ergebnisse an die Clients zurück. Der CONZEPT 16-Server ist also die einzige Instanz, welche die physikalischen Datenbankdateien bearbeiten bzw. Informationen aus ihnen lesen kann. Daher ist der CONZEPT 16-Server auch für die Überwachung, Pflege und Wartung der Datenbanken zuständig.

Die Information, welche Datenbanken der Server zu verwalten hat, entnimmt dieser der sogenannten Datenraumtabelle. Diese Tabelle enthält neben Konfigurationseinträgen auch eine Liste der Datenbanken und deren Speicherort im Dateisystem. Der CONZEPT 16-Server kann dabei beliebig viele Datenbanken mit jeweils bis zu 1000 Tabellen verwalten.

Die wichtigsten Funktionen des Servers im Überblick:

- Verwaltung der Datenbanken
- Datenbankoperationen
- Transaktionsmanagement
- Prozedurale Verarbeitung
- Verwaltung von Sicherungsereignissen
- Funktionen für Datenbankdiagnose, -reparatur und -optimierung
- Funktionen zur Hochverfügbarkeit
- Lizenzmanagement

In den folgenden Kapiteln werden die Funktionsweise und der Betrieb des CONZEPT 16-Servers näher erläutert:

- Architektur
- Installation des Servers
- Administration und Bedienung des Servers
- Hot-Standby

## Kontakt

### **CONZEPT 16-Server - Architektur**

**Genereller Aufbau des CONZEPT 16-Servers und der Datenbanken**

In den folgenden Abschnitten werden der Aufbau des CONZEPT 16-Servers und der CONZEPT 16-Datenbanken erläutert. Dies sind im einzelnen:

- Architektur der Datenbank
- Architektur des Servers
- Speicherverbrauch des Servers
- Transaktionen
- Sicherungsereignisse
- Datenbankdiagnose

## Kontakt

### CONZEPT 16-Server - Architektur der Datenbank

#### Interner Aufbau einer CONZEPT 16-Datenbank

Eine CONZEPT 16-Datenbank besteht aus einem oder mehreren Datenräumen. Ein Datenraum ist eine physikalische Datei, die in Blöcke gleicher Größe, sogenannte Segmente, aufgeteilt ist. Der erste Datenraum (\*.ca1 / Central Area 1) beginnt immer mit einem Header. Innerhalb des Headers sind Informationen über die Anzahl der Datenräume, die Version der Datenbank, sowie Zeitstempel und andere Verwaltungsinformationen abgelegt. Weitere Datenräume werden nur angelegt, wenn diese Trennung manuell vorgenommen wird (beispielsweise aus Datensicherungsgründen).

Eine Datenbank wird durch 2 KB große Segmente unterteilt. Dabei gibt es zwei unterschiedliche Arten von Segmenten. Einerseits Segmente die Daten beinhalten (benutzte Segmente). Diese sind in Baumstrukturen oder Listen organisiert, um einen schnellen Zugriff auf die enthaltenen Daten zu gewährleisten. Und andererseits Segmente ohne Dateninhalte (freie Segmente). Diese entstehen zum Beispiel beim Löschen von Daten. Die freien Segmente werden in Tabellen verwaltet. Um freie Segmente in einer Datenbank zu entfernen und damit die physikalische Datenbankdatei zu verkleinern, kann eine Optimierung der Datenbank durchgeführt werden. Datenräume können aber auch manuell mit freien Segmenten vergrößert werden.

In der Datenbank werden alle Informationen abgelegt, unabhängig davon, ob es sich um Benutzerdaten (z. B. Adressen und Aufträge) oder Applikationsdaten (z. B. Datenstruktur und Programmcode) handelt. Es werden dabei verschiedene Bäume für unterschiedliche Daten in der Datenbank angelegt. Teilweise lassen sich diese Bäume von einer Datenbank in eine andere übertragen. Somit können Applikationsupdates oder Datentransfers ganz einfach durchgeführt werden. Daten in einer Datenbank können im einzelnen sein:

- Datenstruktur
- Prozeduren
- Dialoge
- Datensätze
- Schlüssel
- Texte
- BLOBs (Binary Large Objects)
- Weitere Parameter

Den größten Bereich in einer Datenbank nehmen meist die Datensätze ein. Jeder Datensatz wird mit variabler Länge in die Datenbank geschrieben. Beim Speichern eines Datensatzes wird unabhängig von den in der Datenstruktur definierten Schlüsseln ein eindeutiger Schlüsselwert, die sogenannte Datensatz-ID erzeugt. Alle zugehörigen Schlüsseleinträge verwenden diese ID zur Referenzierung des Datensatzes. Die ID eines Datensatzes bleibt während seiner gesamten Lebensdauer unverändert erhalten. Beim Export von Datensätzen wird die ID nicht mitexportiert, sondern beim Einlesen in eine andere Datenbank wieder neu vergeben. Dies garantiert die Eindeutigkeit der Datensatz-ID, sowie eine sinnvolle Einordnung in die Baumstruktur der Zieldatenbank.

## Kontakt

### CONZEPT 16-Server - Architektur des Servers Aufbau und Funktionsweise des CONZEPT 16-Servers. Betriebshinweise

Der CONZEPT 16-Server kann unter Windows und Linux betrieben werden. Dabei gibt es jeweils zwei Varianten. Zum einen gibt es die 32-Bit Version, welche betriebssystemabhängig bis zu 3 Gigabyte Arbeitsspeicher verwenden kann. Zum anderen gibt es eine 64-Bit Version unter der architekturbedingt praktisch keine Beschränkung des Arbeitsspeichers mehr besteht. Pro verwalteter Datenbank kann der Datenbankserver unter einem 32-Bit Betriebssystem 1 Gigabyte Cache verwenden und unter einem 64-Bit Betriebssystem bis zu 160 Gigabyte.

Der CONZEPT 16-Server kann unter Windows als Dienst oder im Detached-Modus gestartet werden. Wird der CONZEPT 16-Server als Dienst gestartet, initialisiert diesen das Betriebssystem beim Hochfahren automatisch und der CONZEPT

16-Server steht für die Clients und zur Datenbankadministration zur Verfügung. Das Ganze ist somit unabhängig vom Anmeldestatus eines Benutzers. Alternativ kann der CONZEPT 16-Server im aktuellen Benutzerkontext, im sogenannten Detached-Modus, gestartet werden. Der CONZEPT 16-Server startet dann auch ohne Oberfläche, wird aber nicht als Dienst eingerichtet. Unter Linux wird der Server generell als Daemon eingerichtet und läuft somit immer unabhängig von den Benutzer-Sessions.

Die Installation des CONZEPT 16-Servers besteht aus dem Installieren des Lizenzdongle-Treibers und dem Einrichten der CONZEPT 16-Programmdateien. Unter Windows können die Programmdateien und der Dienst über die Setup-Routine eingerichtet und über das CONZEPT 16 Control-Center administriert werden. Unter Linux werden die Programmdateien entpackt und der CONZEPT 16-Server so eingerichtet, dass er automatisch beim Start des Systems initialisiert wird. Um zum Beispiel Statusinformationen von Datenbanken abzufragen oder Sicherungsereignisse zu starten, gibt es zusätzlich noch unter Windows und Linux das kommandozeilenbasierte Script-Utility. Alle Funktionen und Informationen sind zudem auch über den CONZEPT 16-Standard-Client erreichbar.

## Prozesse

Der CONZEPT 16-Server besteht aus drei getrennten Prozesstypen, welche für den Betrieb des Servers verwendet werden. Diese Prozesse werden automatisch gestartet, wenn der CONZEPT 16-Server betrieben wird. Die Prozesstypen teilen sich folgendermaßen auf:

- Service-Prozess

Der Service-Prozess ist der Basisprozess des CONZEPT 16-Servers. Auf ihm bauen alle weiteren Prozesse auf. Daher bildet dieser Prozess auch die Schnittstelle zum Betriebssystem, wenn dieses versucht den CONZEPT 16-Server zu starten oder zu beenden. Auch wenn das Control-Center den CONZEPT 16-Server beenden möchte, wird dazu der Service-Prozess des Datenbankservers angesprochen.

Der Service-Prozess ist nur für Windows und Linux verschieden. Ansonsten ist dieser Prozess der einzige, der bei jeder Rechnerarchitektur und

## Kontakt

Betriebssystemversion der gleiche ist. Bei allen anderen Prozessen wird jeweils automatisch die jeweilige Prozessversion gestartet.

- Manager-Prozess

Der Manager-Prozess ist der zentrale Prozess des Servers. Er wird vom Service-Prozess gestartet und gegebenenfalls auch wieder beendet. Der Manager-Prozess überprüft periodisch die Lizenz des CONZEPT 16-Servers. Wird die Lizenzdatei oder der Lizenzdongle während des Betriebes verändert, registriert dies der Manager-Prozess und verändert die Lizenzdaten des Servers entsprechend. Eine Lizenzerweiterung ist daher auch im laufenden Betrieb möglich. Da der CONZEPT 16-Server auch ohne Lizenz gestartet werden kann, besteht auch die Möglichkeit, die Lizenz erst später einzurichten. Bis zu diesem Zeitpunkt können dann aber keine Datenbanken geöffnet werden.

Der Manager-Prozess führt die Datenraumtabelle des Servers, daher wenden sich alle Clients zunächst an den Manager-Prozess, wenn sie sich an eine Datenbank anmelden möchten. Steht die Datenbank zur Verfügung, verarbeitet der Manager-Prozess diese Anfrage und leitet den Client an den entsprechenden Datenbank-Prozess weiter. Beim Hot-Standby Betrieb von Datenbanken übernimmt der Manager-Prozess zusätzliche Service- und Kommunikationsfunktionen.

- Datenbank-Prozess

Für jede vom Server geöffnete Datenbank gibt es einen Datenbank-Prozess, welcher sich ausschließlich um die Clients kümmert, welche an dieser Datenbank angemeldet sind. Für jeden Client gibt es dabei im Datenbank-Prozess einen eigenen Thread, welcher wiederum nur die Anfragen dieses Clients bearbeitet. Der Datenbank-Prozess ist die einzige Instanz, welche physikalisch auf der Datenbankdatei arbeitet.

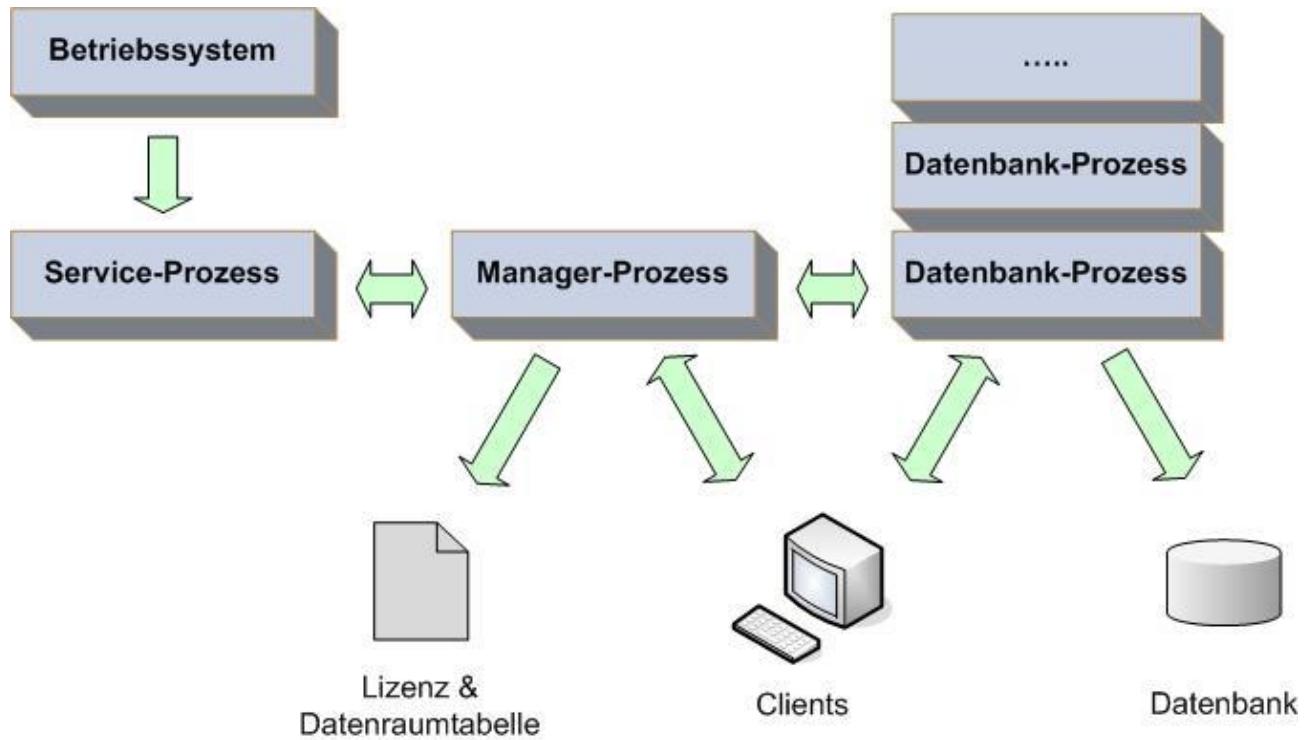
Der Manager-Prozess leitet alle Clients automatisch an die jeweiligen Datenbank-Prozesse weiter, sodass sich diese an die Datenbank anmelden können. Falls eine Datenbank noch geschlossen ist, legt der Manager-Prozess für diese Datenbank zunächst einen eigenen Datenbank-Prozess an, bevor er den Client an diesen weiterleitet. Verlässt der letzte Client die Datenbank, beendet sich der entsprechende Datenbank-Prozess wieder.

Jeder Datenbank-Prozess verwaltet entsprechend der Konfiguration seinen eigenen Datenbankcache. Die Anzahl der Datenbanken, welche geöffnet werden können, hängt daher nur vom physikalisch verfügbaren Arbeitsspeicher des Systems ab. Durch die Unabhängigkeit der Prozesse können sich Abläufe verschiedener Datenbank-Prozesse gegenseitig nicht beeinflussen, was wiederum die Stabilität des Datenbankservers erhöht.



Da alle Server-Prozesse unabhängig voneinander laufen, gibt es generell keine wechselseitigen Abhängigkeiten zwischen den Prozessen. Wird zum Beispiel der Manager-Prozess beendet, können sich zwar keine neuen Benutzer an die Datenbanken anmelden, die bestehenden können aber trotzdem weiterarbeiten.

## Kontakt



### Temporäre Dateien

Der Datenbank-Prozess legt eine temporäre Datei mit dem Namen <Datenbankname>.tmp im eingestellten Transaktionsverzeichnis der Datenbank ab (siehe auch Konfiguration der Datenbanken). Dort werden Daten abgelegt, die beispielsweise während einer Prozedurdurchführung auf dem Server anfallen.

### Log-Dateien

#### *Serverlogs*

Alle Prozesse notieren Informationen, Warnungen und Fehler in Form von Log-Einträgen in verschiedenen Log-Dateien. Die Log-Dateien des Service- und des Manager-Prozesses heißen c16\_serv\_svc.lgb und c16\_serv\_mgr.lgb und liegen im Standardverzeichnis für Konfigurationsdateien. Unter Linux-Systemen werden diese Dateien direkt im Programmverzeichnis des Datenbank-Servers abgelegt. Für jede Datenbank verwenden die Datenbank-Prozesse eine eigene Log-Datei mit dem Namen "Datenbankname".lgb. Diese liegt im Verzeichnis der Datenbank. Informationen zu den verschiedenen Log-Einträgen können unter Log-Einträge entnommen werden.

#### *Benutzerlogs*

Der CONZEPT 16-Server kann auf Wunsch auch eine Benutzerlog-Datei anlegen und diese mit benutzerspezifischen Log-Einträgen füllen. Diese Einträge können über den Befehl DbaLog() in der Log-Datei abgelegt werden. Benutzerlogs werden im Datenbankverzeichnis unter "Datenbankname".lgu abgelegt.

#### *Log-Format*

## Kontakt

Die Log-Dateien werden in einem binären Format geschrieben und sind daher manipulationssicher. Sie können mit dem Log-Viewer und dem Script Utility ausgelesen werden. Die Dateien sind auf eine Größe von 8 MB beschränkt und damit auch beispielsweise für einen Versand per E-Mail geeignet. Alle Log-Einträge werden chronologisch in den jeweiligen Log-Dateien gesichert. Kommen bei einer Maximalgröße neue Einträge dazu, werden die ältesten entfernt.

Ausnahmen bilden die Log-Dateien der Datenbanken und die Benutzer-Logs. Bei diesen Log-Formaten gibt es neben den Einträgen des Systems auch Einträge, die sich auf eine Benutzerinteraktion beziehen, wie beispielsweise ein Login in eine Datenbank. Den in der Regel weniger auftretenden Systemeinträgen wird allerdings ein höherer Stellenwert zugeschrieben, da sie meist für eine längere Zeit nachvollziehbar sein müssen. Aus diesem Grund ist für Systemeinträge immer mindestens die Hälfte der Maximalgröße eines Logs reserviert. Nur wenn diese Grenze erreicht wird und neue Systemeinträge hinzukommen, werden die älteren Systemeinträge aus dem Log entfernt. In allen anderen Fällen entfernt der CONZEPT 16-Server die ältesten Benutzereinträge.

Die Logs der Datenbank und der Benutzer besitzen noch eine weitere Besonderheit. Entfernte Einträge werden nicht verworfen sondern landen jeweils in einem chronologischen Archivlog, welches auf eine Größe von 128 MB beschränkt ist. Der Dateiname des Archivlogs entspricht dabei dem Dateinamen des normalen Logs, mit der Erweiterung \*.lgbx beziehungsweise \*.lgux. In der Regel reicht das normale Log völlig aus. Nur für den Fall, dass Log-Einträge über mehrere Jahre zurück benötigt werden, kann der Administrator über den Log-Viewer bis in ein eventuell vorhandenes Archivlog zurückblättern. Einträge, die aufgrund der Dateimaximalgröße aus dem Archivlog entfernt werden, sind endgültig gelöscht.

## Netzwerkprotokolle

Der CONZEPT 16-Server kommuniziert üblicherweise nur über das TCP/IP-Protokoll. Werden Datenbankserver und Clients allerdings auf dem gleichen System betrieben, kann optional auch ein für den lokalen Betrieb optimiertes Protokoll verwendet werden. In diesem Fall wird nicht die IP-Nummer oder der Name des Servers angegeben, mit dem man sich verbinden möchte, sondern ein \*. Besonders für Job-Server oder ähnlichem ist dies sehr sinnvoll, wenn diese auf dem Serversystem betrieben werden, da durch dieses Alternativprotokoll der Datendurchsatz drastisch erhöht werden kann.

## TCP/IP-Ports

Die folgenden TCP/IP-Ports werden vom CONZEPT 16-Server während des Betriebes verwendet. Um die jeweiligen Funktionen über das Netzwerk erreichbar zu machen, müssen daher die entsprechenden Ports freigegeben werden.

**4722** Auf dem Port 4722 nimmt der Manager-Prozess die eingehenden Datenbankanfragen von den Clients entgegen und gibt sie an die Datenbank-Prozesse weiter. Die Datenbank-Prozesse verwenden für die Kommunikation mit den Clients ebenfalls den Port 4722.

## Kontakt

Auf dem Port 4741 wird die gesamte interne Kommunikation des Servers geregelt. Das bedeutet, dass der Service-Prozess, der Manager-Prozess und die Datenbank-Prozesse alle zum Betrieb notwendigen Daten über diesen Port austauschen. Da diese Kommunikation lokal stattfindet, gibt es für diesen Port keinen Anwendungsfall, für den er freigegeben werden müsste.

4742 Für die Kommunikation der Server-Prozesse mit dem Script-Utility und dem Control-Center wird der Port 4742 verwendet.

4743 Der Port 4743 wird für die Hot-Standby Kommunikation verwendet. Hierüber wird zum Beispiel der Hot-Standby Betrieb einer Datenbank initiiert.

4745 Dies ist der voreingestellte Port zur Administration des Servers über einen Browser. Er kann in der Konfiguration des Servers geändert werden.

4751 Der Port 4751 wird für die Kommunikation des SOA-Service mit den Tasks verwendet. Da diese Kommunikation lokal stattfindet, gibt es für diesen Port keinen Anwendungsfall, für den er freigegeben werden müsste.

4752 Für die Kommunikation des SOA-Service mit dem Script-Utility (SOA-Service) und dem Control-Center wird der Port 4752 verwendet.

## Kontakt

### **CONZEPT 16-Server - Speicherverbrauch des Servers Beschreibung des Speicherverbrauchs des CONZEPT 16-Servers**

Dem Bedarf an Hauptspeicher für den CONZEPT 16-Server ist besonderes Augenmerk zu widmen. Maximal 75% des zur Verfügung stehenden physikalischen Hauptspeichers darf vom CONZEPT 16-Server benutzt werden, empfehlenswert sind allerdings maximal 50%. Wird der Serverrechner noch für andere Zwecke intensiv benutzt, muss eine entsprechend verbleibende Speichergröße einkalkuliert werden (siehe auch Hardware-Empfehlung für den Datenbank-Server).

Der CONZEPT 16-Server besteht aus mehreren Prozessen (siehe Architektur des Servers) von denen hier nur der Datenbank-Prozess relevant ist. Der Speicherverbrauch setzt sich aus folgenden Elementen zusammen:

- Statischer Speicher Pro Datenbank-Prozess werden ca. 4 MB Hauptspeicher benötigt.
- Datenbank-Cache Der Datenbank-Cache kann vom Administrator konfiguriert werden (siehe Konfiguration der Datenbanken). Jede Datenbank benötigt für ein performantes Transaktionsmanagement einen Cache, der mindestens 10% der Datenbankgröße entspricht. Zudem werden pro Benutzer in der Datenbank auch mindestens 100 KB Cache benötigt. Der tatsächliche gesamte Speicherbedarf liegt um ca. 10% höher als der errechnete, da zum Cache zusätzlich weitere Verwaltungsstrukturen notwendig sind.
- Speicher pro Benutzer Für jeden in der Datenbank angemeldeten Benutzer werden zusätzlich zum Cache weitere 100 KB Hauptspeicher benötigt.
- Diagnose / Recover / Optimierung Bei der Durchführung einer dieser Funktionen sind 10 KB Hauptspeicher pro MB Datenbank erforderlich. Bei einer Datenbank von 10 GB werden also temporär 100 MB Hauptspeicher benötigt.

Der hier bestimmte Speicherbedarf muss für alle gleichzeitig geöffneten Datenbanken berechnet und addiert werden. Es ist zu beachten, dass abhängig vom verwendeten Betriebssystem die Grenzen für den maximalen virtuellen Speicher pro Prozess nicht überschritten werden.



Wenn der CONZEPT 16-Standard-Client an einer Datenbank angemeldet ist, besteht die Möglichkeit sich den aktuellen Speicherverbrauch des entsprechenden Datenbank-Prozesses, sowie den bisherigen Maximalverbrauch anzeigen zu lassen. Dazu muss der Menüeintrag Info / Server aufgerufen werden.

In den 32-Bit-Versionen der Windows-Betriebssysteme stehen für jeden Prozess maximal 4 GB virtueller Hauptspeicher zur Verfügung. Dieser Speicher wird zur Hälfte vom Prozess selbst und zur anderen Hälfte vom Betriebssystem verwendet. Einem Prozess stehen so maximal 2 GB Hauptspeicher zur Verfügung. Bei folgenden Betriebssystemen kann durch die Angabe des Schalters /3GB in der Datei Boot.ini die Aufteilung des virtuellen Speichers zu Gunsten des Prozesses verschoben werden:

- Windows 2003, Datacenter Edition
- Windows 2003, Enterprise Edition

## Kontakt

- Windows 2003
- Windows XP Professional
- Windows Server 2008
- Windows Server 2008 R2
- Windows Vista
- Windows 7
- Windows 8
- Windows 8.1
- Windows 10



Bei Betriebssystemen ab Windows Vista existiert die Boot.ini nicht mehr. Dort muss der Schalter mit dem Kommando BCDEdit angelegt werden. Nähere Informationen finden Sie in dem [MSDN-Artikel](#) von Microsoft.

Unter Linux ist dies nicht möglich, da hier generell 3 GB für die Prozesse und 1 GB für das Betriebssystem verwendet werden. Nähere Informationen zu diesem Thema befinden sich in der Dokumentation des jeweiligen Betriebssystems.



Bei Systemen mit 4 GB und mehr Hauptspeicher sollten 64-bit Betriebssysteme eingesetzt werden.

## Kontakt

### CONZEPT 16-Server - Transaktionen Beschreibung der Transaktionsverarbeitung

Transaktionen sind eine zusammengehörige Menge von Datenbankoperationen. Diese werden zunächst im Datenbankcache verarbeitet. Den Transaktionen werden die Inhalte des ACID-Prinzips zugrunde gelegt.

Atomicity	Atomizität	Eine Transaktion wird vollständig in der Datenbank gespeichert oder gar nicht.
Consistency	Konsistenz	Nach der Transaktion ist der Datenbestand widerspruchsfrei.
Isolation	Isolation	Mehrere Transaktionen beeinflussen sich nicht gegenseitig.
Durability	Dauerhaftigkeit	Eine beendete Transaktion bleibt in der Datenbank bestehen.

### Transaktionsressourcen

Der Datenbank-Prozess nutzt eine Reihe von Ressourcen, um den Datenbankbetrieb und damit auch das Transaktionsmanagement korrekt durchführen zu können. Zunächst fordert jeder Datenbank-Prozess die konfigurierte Menge an Datenbankcache an. Der Cache wird verwendet, um Datenbanksegmente zwischenzuspeichern und die Zugriffszeiten darauf zu verkürzen. Zudem verarbeitet der Cache die Datenbanktransaktionen. Dabei werden laufende und abgeschlossene Transaktionen im Cache vorgehalten. Sollte der Cache für die Transaktionen nicht ausreichen, lagert der Datenbank-Prozess Informationen in einer temporären Transaktionsdatei mit dem Namen <Datenbankname>.trs aus.

### Transaktionsverklemmungen

Wenn der Inhalt einer Datenbank verändert wird, findet dies immer auf Segment-Ebene statt. Eine Transaktion setzt sich aus einer zusammengehörigen Menge von geänderten Segmenten zusammen. Falls zwei Benutzer verschiedene Segmente durch ihre Transaktion sperren und jeweils auf die Freigabe der Segmente des anderen warten, entsteht eine Verklemmung. Diese Verklemmungen erkennt der CONZEPT 16-Server automatisch und löst sie auf. Dazu wird eine Transaktion durchgeführt und die andere abgebrochen. Abgebrochen wird dabei immer die kleinere Transaktion. Entsteht keine Verklemmung, sondern nur ein Wartezustand (Sperrkonflikt), zeigt dies der CONZEPT 16-Client dem Anwender an.

### Ablauf von Transaktionen

Offene Transaktionen befinden sich immer im Datenbankcache oder in der ausgelagerten temporären Transaktionsdatei. Abgeschlossene Transaktionen werden mindestens alle 30 Sekunden in die Datenbank übertragen. Falls der Datenbankcache zu 80% mit abgeschlossenen Transaktionen gefüllt ist, bevor die 30 Sekunden vergangen sind, wird das Übertragen der abgeschlossenen Transaktionen vorgezogen.

Das Schreiben von abgeschlossenen Transaktionen in die Datenbank wird Update-Ereignis genannt. Ein Update-Ereignis wird in mehrere zeitlich versetzte Schritte untergliedert, um zu jeder Zeit die Konsistenz der Datenbank zu gewährleisten. Beim Ausführen des Update-Ereignisses dürfen die abgeschlossenen

## Kontakt

Transaktionen nicht einfach in die Datenbank übertragen werden, da ein Ausfall des Servers während der Übertragung die Datenbank in einen inkonsistenten Zustand bringen würde. Abgeschlossene Transaktionen könnten in diesem Fall nur teilweise und damit fehlerhaft in die Datenbank gelangt sein.

Um dies zu verhindern, werden alle Segmente eines Update-Ereignisses zunächst extern in eine Transaktionslog-Datei geschrieben (<Datenbankname>.tl1). Dies ist notwendig, damit alle abgeschlossenen Transaktionen auf der Festplatte gesichert sind. Erst danach findet das Übertragen der Transaktionen in die Datenbank statt. Fällt dabei der Server aus, kann aus den Segmenten in der externen Transaktionslog-Datei die Konsistenz der Datenbank nachträglich wiederhergestellt werden und die Daten sind nicht verloren. Der CONZEPT 16-Server sichert dabei zyklisch die letzten drei Transaktionslog-Dateien (\*.tl1, \*.tl2, \*.tl3).

## Rollback

Falls der Datenbankserver im laufenden Datenbankbetrieb ausfällt, kann dieser die Datenbank nicht mehr korrekt schließen. Das bedeutet, dass die abgeschlossenen Transaktionen des letzten Update-Ereignisses möglicherweise noch nicht oder nicht komplett in die Datenbank übertragen wurden.

Nun muss vom Datenbankserver sichergestellt werden, dass die Datenbank in einem konsistenten Zustand ist. Dazu wird beim nächsten Öffnen der Datenbank die letzte Transaktionslog-Datei in die Datenbank übertragen. Dadurch werden alle abgeschlossenen Transaktionen des letzten Update-Ereignisses nochmals korrekt in die Datenbank übernommen. Alle Daten des letzten Update-Ereignisses werden so wiederhergestellt und die Datenbank ist wieder in einem konsistenten Zustand.

Falls der Datenbankserver ausfällt, während die Transaktionslog-Datei geschrieben wird, liest der CONZEPT 16-Server diese nicht ein, da sie möglicherweise unvollständige Transaktionen enthält. Alle Daten des vorherigen Update-Ereignisses befinden sich bereits in der Datenbank, da dies vom Datenbankserver sichergestellt wird, bevor das nächste Update-Ereignis ausgelöst wird.

Generell muss beachtet werden, dass alle offenen und auch alle abgeschlossenen Transaktionen verloren gehen, die sich zum Zeitpunkt des Serverausfalls im Datenbankcache und der ausgelagerten, temporären Transaktionsdatei befinden.

-  Ein Rollback kann nur vorgenommen werden, wenn die Transaktionslog-Dateien in der Konfiguration der Datenbank aktiviert sind. Nach dem Rollback wird automatisch eine Diagnose mit Recover gestartet.
-  Werden nach einem Serverabsturz Fehler im Dateisystem festgestellt, müssen diese vor einem erneuten Öffnen der Datenbank beseitigt werden, damit Probleme beim Rollback durch äußere Einwirkungen minimiert werden können. Beim Betrieb von Festplatten, bzw. RAID-Controllern mit verzögertem Schreiben, also eigenem Schreib-Cache, muss sichergestellt werden, dass keine Datenverluste an den jeweiligen Stellen entstehen können (zum Beispiel über Batteriebackups oder eine unterbrechungsfreie Stromversorgung (USV)).

## Transaktionen während Sicherungsereignissen

## Kontakt

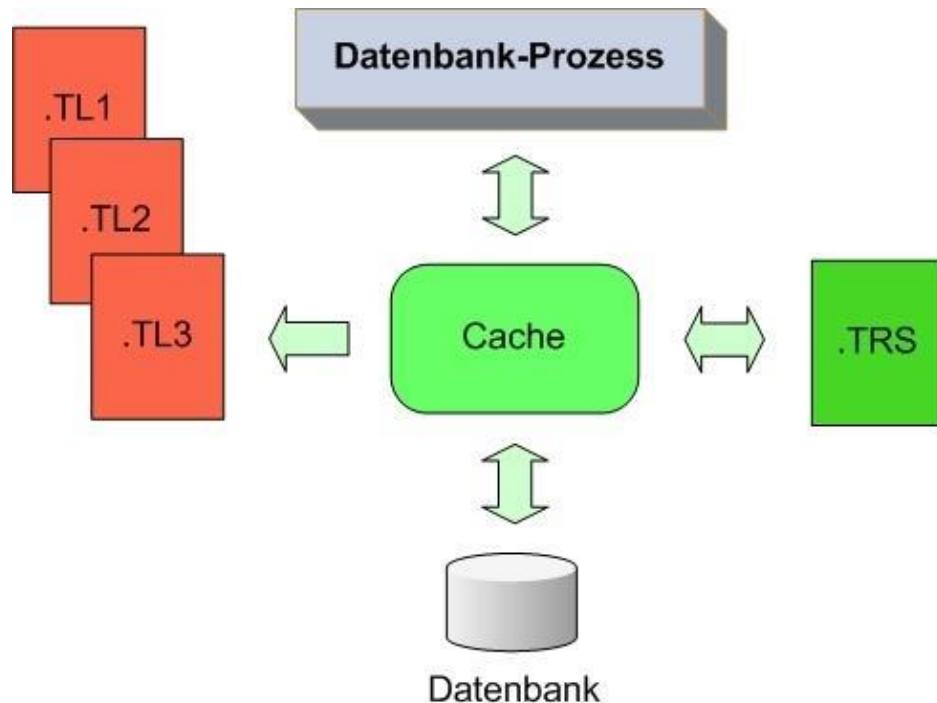
Der CONZEPT 16-Server kann Datenbanken in einen Sicherungszustand versetzen. (Siehe Sicherungssereignisse) Dazu wird die Datenbank im Read-Only Modus betrieben. Alle in dieser Zeit abgeschlossenen Transaktionen können dann natürlich nicht direkt in die Datenbank übertragen werden.

Während eines Backup-Ereignisses werden keine Update-Ereignisse durchgeführt. Alle abgeschlossenen Transaktionen werden im Datenbankcache und der temporären Transaktionsdatei gesichert. Während eines Backup-Ereignisses werden trotzdem wie bei einem Update-Ereignis alle abgeschlossenen Transaktionen in eine Transaktionslog-Datei geschrieben, damit bei einem Serverausfall die Daten wiederhergestellt werden können. Hierbei wird aber fortlaufend nur in die aktuelle Transaktionslog-Datei geschrieben. Ein zyklischer Wechsel findet nicht statt, da sonst nach einem Zyklus Transaktionslog-Dateien überschrieben werden.



Zu Beachten ist daher natürlich auch, dass die aktuelle Transaktionslog-Datei und die temporäre Transaktionsdatei bei längeren Backup-Ereignissen sehr groß werden können.

Nach dem Backup-Ereignis werden alle abgeschlossenen Transaktionen aus dem Cache und der temporären Transaktionsdatei in die Datenbank geschrieben. Falls der Server nun während des Sicherungssereignisses abstürzt, würde unter normalen Umständen kein Rollback durchgeführt werden, da die Datenbank in einem korrekt geschlossenen Zustand vorliegt. Die Transaktionslog-Dateien enthalten den Updatezähler und Zeitstempel der Datenbank, sowie eine Seriennummer. Stimmen der Zeitstempel und der Updatezähler der Datenbank mit denen der Transaktionslog-Datei überein und hat diese die Seriennummer 1, muss diese noch in die Datenbank eingelesen werden. Wird beim Öffnen der Datenbank eine solche Datei vorgefunden, erkennt der Datenbankserver, dass ein Backup-Ereignis nicht korrekt beendet wurde und noch abgeschlossene Transaktionen aus der letzten Transaktionslog-Datei in die Datenbank eingefügt werden müssen.



## Kontakt



**Alle temporären Dateien werden standardmäßig im Transaktionsverzeichnis der Datenbank abgelegt. In der Datenbankkonfiguration kann dieses Verzeichnis angegeben werden. Wird dieses Verzeichnis nicht angegeben, ist das Verzeichnis der Datenbank auch zugleich das Transaktionsverzeichnis.**

## Kontakt

### Transaktionsverarbeitung

#### Verarbeitungen von Transaktionen

Segmente, die während einer Transaktion verändert werden, bleiben im Datenbankpuffer bis die Transaktion beendet ist. Bei der Verdrängung solcher Segmente aus dem Puffer werden diese in die temporäre Transaktionsdatei übertragen. Jede Änderung in der Datenbank wird innerhalb einer Transaktion definiert, auch wenn keine expliziten Transaktionen in der Applikation benutzt werden.



Änderungen an den folgenden Datenbankinhalten werden von der Transaktion berücksichtigt:

- Datensätze
- Prozeduren
- Texte
- Binäre Verzeichnisse und Objekte
- Dialog-Objekte
- Bilder in der Ressourcenverwaltung
- Selektionen



Kehrt eine Prozedur in den Designer zurück oder wird ein Benutzer abgemeldet, werden die offenen Transaktionen abgebrochen des Benutzers.

#### Sperrkonflikte

Transaktionssegmente bleiben bis zum Ende der Transaktion gesperrt, das heißt andere Benutzer können Daten im gleichen Segment nicht verändern. Beim Versuch eines Benutzers, Daten in einem Segment zu ändern, das bereits durch eine offene Transaktion eines anderen Benutzers geändert wurde, entsteht ein sogenannter Sperrkonflikt. Dieser führt zum Warten des Benutzers auf das Transaktionsende und die damit verbundene Freigabe des Segments. Dauert der Wartezustand länger als zwei Sekunden, wird auf der Station entweder ein "W" in der obersten Zeile angezeigt oder es erscheint die Meldung "Bitte warten...".

Das Meldungsfenster kann über die Eigenschaften Caption und CaptionColor des App-Objekts angepasst werden.

#### Verklemmungen

Sperrkonflikte können zu Verklemmungen (Deadlocks) führen, wenn beispielsweise Benutzer A auf ein Segment von Benutzer B wartet und Benutzer B auf ein Segment von Benutzer A. In diesem Fall würden beide Benutzer ewig warten. Der CONZEPT 16-Server stellt eine solche Situation fest und bricht eine der beteiligten Transaktionen ab. Dabei wird die Transaktion abgebrochen, in der bisher am wenigsten Datenbankoperationen durchgeführt wurden.

## Kontakt

### CONZEPT 16-Server - Sicherungsereignisse

#### Beschreibung der Sicherungsereignisse des CONZEPT 16-Servers

##### Sicherungsereignisse

Datenbanken sollten in regelmäßigen Abständen gesichert werden. Auch vor und nach größeren Eingriffen in den Datenbestand ist es unter Umständen sinnvoll eine zusätzliche Sicherung der Datenbank durchzuführen. Um eine Datenbank zu sichern müssen nur die physikalischen Datenbankdateien (<Datenbankname>.ca?) gesichert werden. Eine Sicherung der Log-Dateien ist sinnvoll, aber nicht notwendig. Die Transaktions-Log Dateien, sowie die temporäre Transaktionsdatei und die temporäre Datei des Datenbank-Prozesses müssen nicht gesichert werden. Im laufenden Betrieb der Datenbank ist dies zudem auch nicht möglich.

Datenbanken die beim CONZEPT 16-Server eingetragen sind, können grundsätzlich frei kopiert werden. Unter Windows sperrt der Datenbankserver die Datenbank aber exklusiv, wenn diese geöffnet wird. Das bedeutet, dass dann nur noch der CONZEPT 16-Server auf die Datenbank zugreifen kann. Unter Linux ist ein Kopieren aus Gründen des Dateisystems immer möglich. Ein Kopieren im laufenden Betrieb ist aber nicht sinnvoll, da sonst ein inkonsistenter Zustand der Datenbank gesichert wird. Haben alle Benutzer die Datenbank wieder verlassen, wird der exklusive Zugriff auf die physikalische Datenbankdatei wieder aufgelöst und die Datenbank kann wieder kopiert werden.



Wird eine zum Lesen und Schreiben geöffnete Datenbank gesichert, ist die Sicherung inkonsistent. Datenbanken müssen während eines Sicherungsereignisses kopiert werden.

Sicherungsereignisse (Backup-Ereignisse) ermöglichen ein Sichern von offenen Datenbanken mit angemeldeten Benutzern. Dazu wird die Datenbank vom CONZEPT 16-Server zeitweise im Read-Only Modus betrieben. Die Datenbank ist dann trotz der angemeldeten Benutzer kopierbar. Änderungen am Datenbestand werden während des Backup-Ereignisses gesichert und danach in die Datenbank geschrieben.

#### Ablauf eines Sicherungsereignisses

Beim Auslösen eines Backup-Ereignisses wird zunächst ein letztes Update-Ereignis ausgeführt und alle abgeschlossenen Transaktionen in die Datenbank geschrieben. Danach schließt der CONZEPT 16-Server die physikalische Datenbankdatei und öffnet sie im Read-Only Modus.

Die Datenbankdateien können nun während des Backup-Ereignisses kopiert werden. Update-Ereignisse werden während eines Backup-Ereignisses nicht durchgeführt. Alle abgeschlossenen Transaktionen werden im Datenbankcache und in der temporären Transaktionsdatei gesichert. Damit bei einem Serverausfall nicht alle abgeschlossenen Transaktionen verloren gehen, werden analog zum Update-Ereignis abgeschlossene Transaktionen in die aktuelle Transaktionslog-Datei geschrieben (siehe Transaktionen während Sicherungsereignissen).

Nach einem Backup-Ereignis wird die Datenbank wieder im exklusiven Zugriff geöffnet. Falls das nicht gelingt, versucht dies der CONZEPT 16-Server wiederholt im Sekundenabstand. Wenn die Datenbank wieder exklusiv geöffnet werden kann,

## Kontakt

werden alle abgeschlossenen Transaktionen aus dem Datenbankcache und der temporären Transaktionsdatei in die Datenbank geschrieben. Erst dann ist das Backup-Ereignis korrekt beendet.

-  Um eine Datenbank auf ein Medium zu sichern, welches für den Sicherungsvorgang sehr lange benötigt, ist es sinnvoll die Datenbank zunächst einfach zu kopieren und danach mit der Kopie die Datensicherung durchzuführen. Somit ist die Datenbank nicht zu lange im Backup-Zustand. Das Initiieren des Backup-Zustandes und das anschließende Kopieren der Datenbank, sowie das Rücksetzen des Backup-Zustandes kann zum Beispiel über ein Skript durchgeführt werden (siehe nächster Abschnitt, Auslösen eines Sicherungseignisses).

### Auslösen eines Sicherungseignisses

Die Zeitpunkte von Backup-Ereignissen werden wie andere Konfigurationseinstellungen in der Datenraumtabelle des Servers festgehalten ([Konfiguration des Servers](#)). Das Starten eines Backup-Ereignisses per Prozedur ist auch möglich ([DbaControl\(\)](#)). Das Backup-Ereignis kann zudem mit Hilfe des [Script-Utilitys](#) von der Konsole gestartet und gestoppt werden.

-  Solange eine Diagnose mit Recover oder Optimierung der Datenbank läuft, kann kein Backup-Ereignis gestartet werden. Ist eine Datenbank als Slave-Datenbank geöffnet (siehe [Hot-Standby](#)), können auch keine Backup-Ereignisse gestartet werden.

Vom [Volumenschattenkopiedienst](#) von Windows werden die definierten Datenbanken (siehe [Konfiguration des Servers](#)) automatisch in den Backup-Zustand versetzt, wenn sie geöffnet sind oder werden. Eine Slave-Datenbank kann in diesem Fall jedoch von der Volumenschattenkopie erfasst werden. Eine Umschaltung in den Read-Only Modus findet hier jedoch nicht statt.

## Kontakt

**Volumenschattenkopiedienst**

**Schattenkopien von CONZEPT 16-Datenbanken**

**Siehe Blog**

**Was ist der Volumenschattenkopiedienst?**

**Der Volumenschattenkopiedienst (engl. Volume Shadow Copy Service, Abk. VSS) ist ein Dienst von Windows, der Momentaufnahmen von einem oder mehreren Massenspeichern bereitstellt. Diese Schattenkopien erlauben beispielsweise die Datensicherung im laufenden Betrieb.**

**Wie funktioniert der Volumenschattenkopiedienst?**

**Snapshots werden im laufenden Betrieb erzeugt, indem ab dem Erstellungszeitpunkt alle Schreiboperationen auf dem Volume entweder in einen alternativen Speicherbereich geschrieben werden (Redirect-on-Write) oder der ursprüngliche Inhalt des jeweiligen Speicherblocks in den Speicherbereich des Snapshot gesichert wird (Copy-on-Write). Damit der Snapshot brauchbare Daten enthält, muss das Volume zum Erstellungszeitpunkt einen konsistenten Zustand aufweisen. Dateisystem, Metadaten, System- und Anwendungsdatenbanken müssen daher vor der Erstellung des Snapshots entsprechend präpariert werden.**

**Um eine Schattenkopie zu erstellen, benötigt man drei Komponenten:**

- Provider
- Requester
- Writer

*Provider*

Ein "Provider" stellt die Funktionen zur Generierung, Verwaltung und Nutzung von Snapshots bereit. Standardmäßig ist ein "System Provider" im Windows-Betriebssystem vorhanden, der Schattenkopien auf der Ebene des Dateisystems (NTFS oder ReFS) realisiert. Zusätzlich lassen sich weitere Provider installieren, die die Snapshot-Funktionalität entweder in der Schicht der Storage-Treiber ("Software Provider") oder in der Storage-Hardware ("Hardware Provider"), beispielsweise im Controller oder in einem SAN-System, abbilden.

*Requester*

Ein "Requester" ist üblicherweise ein Backup-Client oder -Agent. Er gibt beim "Provider" eine Schattenkopie von ausgewählten oder allen Datenträgern in Auftrag. Nachdem die Daten von der Schattenkopie zum Sicherungsziel kopiert wurden, beauftragt der "Requester" den "Provider" die Schattenkopie wieder zu entfernen.



**Nicht jede Backup-Software unterstützt auch VSS. Besonders auf Client-Betriebssystemen ist die VSS-Unterstützung selten vorhanden.**

*Writer*

Ein "Writer" wird zur Herstellung eines konsistenten Zustandes benötigt. Er übernimmt die Aufgabe die Dateien seiner Anwendung einzufrieren und nach erfolgter

## Kontakt

Erstellung der Schattenkopie (nicht nach der Kopie der Daten) wieder in den Normalzustand zu versetzen. Für CONZEPT 16-Datenbanken ist somit ein spezieller VSS-Writer erforderlich.

### VSS in CONZEPT 16

Im CONZEPT 16-Server ist ein VSS-Writer realisiert, der die ausgewählten Datenbanken (siehe Konfiguration des Servers) automatisch in den Backup-Modus versetzt, wenn dies von einem Requester angesteuert wird. Der VSS-Writer ist standardmäßig für alle Datenbanken aktiviert.

Nachdem die Datenbanken eingefroren sind, besteht ein Timeout von 5 Minuten, um die Datenbanken wieder in den Normalzustand zu überführen. Wird dieses Timeout überschritten, wird der Backup-Zustand beendet und der Eintrag VSS Backup event completed (timeout) in der Log-Datei der Datenbank protokolliert.

Befindet sich eine Datenbank bereits im Backup-Modus, wird sie erst wieder in den Normalzustand versetzt, wenn der Backup-Modus, der länger andauert, beendet wurde.

Datenbanken, die als "Nur Lesen" (siehe Konfiguration des Servers) geöffnet wurden, werden nicht in den Backup-Modus versetzt und sind nicht von VSS betroffen.



Zum Betrieb des CONZEPT 16-VSS-Writers wird mindestens Windows Vista bzw. Windows Server 2003 benötigt.

#### Wiederherstellung von Sicherungen

Soll eine Datenbank in einem früheren Zustand wiederhergestellt werden, müssen die folgenden Bedingungen erfüllt sein:

- *Der CONZEPT 16-Server ist als Dienst gestartet*

Der Dienst wird bei der Wiederherstellung über VSS automatisch beendet und nach der Wiederherstellung gestartet. Offene Datenbanken werden automatisch geschlossen.

- *Der CONZEPT 16-Server ist Detached gestartet*

Betroffene Datenbanken müssen manuell geschlossen werden, bevor sie wiederhergestellt werden können.

#### Vorteile von VSS

- kurze Sicherungsereignisse

Die Datenbank wird nur für den Zeitpunkt der Erstellung der Schattenkopie in den Backup-Modus versetzt. Während der eigentlichen Sicherung befindet sich die Datenbank im Normalzustand. Das Backup-Ereignis dauert meist nur wenige Sekunden.

- weniger Administrationsaufwand und geringere Fehleranfälligkeit

## Kontakt

Es besteht keine Notwendigkeit, Backup-Zeiten einzutragen oder das Script-Utility einzubinden, die Steuerung der Backup-Ereignisse übernimmt automatisch die Sicherungssoftware. Die Gefahr von fehlenden oder nicht zu den Sicherungen passenden Backup-Ereignissen entfällt.

- Unterstützung komplexer Lösungen

Neben klassischer Backup-Software wird VSS auch von speziellen Data-Protection-Produkten unterstützt. Dazu zählen beispielsweise Snapshot- und Replikationsanwendungen, die direkt auf den Storage-Systemen laufen, Sicherungsanwendungen auf der Virtualisierungsebene oder Anwendungen wie der "System Center Data Protection Manager".

- Sicherung der Standby-Datenbank möglich

Ist eine Datenbank als Slave-Datenbank geöffnet (siehe Hot-Standby), wird die Datenbank in einen konsistenten Zustand gebracht. Eine Umschaltung in den Read-Only Modus erfolgt in diesem Fall jedoch nicht.

## Kontakt

### CONZEPT 16-Server - Datenbankdiagnose

#### Beschreibung der Diagnose von CONZEPT 16-Datenbanken

##### Erweiterte

Siehe [Diagnose](#)

[\(Blog\)](#)

Der CONZEPT 16-Datenbankserver verfügt über Diagnosefunktionen zur Überprüfung und Reparatur von Datenbankstrukturen. Eine regelmäßige Diagnose ist sinnvoll, um möglichst frühzeitig auftretende Fehler erkennen und korrigieren zu können.

Eine Datenbankdiagnose überprüft die Struktur der in der Datenbank gespeicherten Daten auf Fehler. Jede Datenbankdiagnose erzeugt ein Diagnoseresultat, aus welchem ermittelt werden kann, ob und an welchen Stellen Fehler in der Datenbankstruktur festgestellt wurden. Eine normale Datenbankdiagnose nimmt keine Veränderung an der Datenbank vor. Fehler werden dabei nur protokolliert, nicht behoben. Eine Diagnose kann während des laufenden Betriebs durchgeführt werden. Die Datenbank wird während der Diagnose in einen Backup-Zustand versetzt (siehe [Sicherungereignisse](#))

Die Datenbankdiagnose benötigt ca. 10 KB Hauptspeicher pro MB Datenbank auf dem Server. Bei einer Datenbankgröße von 10 GB werden also ca. 100 MB Hauptspeicher benötigt. Dieser Speicher wird zusätzlich zum Datenbankcache benötigt. Die normale Datenbankdiagnose kann zusätzlich noch um einige Optionen erweitert werden.



Falls eine Diagnose mit "Recover" oder eine "Schlüsselanalyse und -reparatur" durchgeführt wird, müssen möglicherweise Änderungen an der Datenbank vorgenommen werden. Vor der Diagnose muss daher unbedingt eine Sicherung der Datenbank vorgenommen werden, da ein Ausfall des Servers während der Diagnose zu Datenverlust führen kann.

- **Diagnoseoption "Erweiterte Diagnose"**

Die erweiterte Datenbankdiagnose verhält sich genauso wie die normale Diagnose. Die einzige Ausnahme ist, dass die erweiterte Diagnose die Datenbank auch auf seltene Sortierungsprobleme innerhalb der Baumstruktur der Datenbank hin untersucht. Dabei ist zu beachten, dass diese Diagnose unter Umständen ein Vielfaches der Zeit der normalen Diagnose beanspruchen kann.

Bei größeren Datenbanken ist es sinnvoll die erweiterte Diagnose zunächst auf einer Kopie der Datenbank durchzuführen, damit es nicht zu einem längeren Backup-Zustand der Original-Datenbank kommt.

- **Diagnoseoption "Recover durchführen"**

Bei der Datenbankdiagnose mit Recover müssen alle Benutzer die Datenbank verlassen und die Datenbank darf sich nicht im Backup-Modus befinden, da der Diagnoselauf exklusive Zugriffsrechte auf die Datenbank benötigt.

Bei der Diagnose mit Recover werden zum einen Überprüfungen durchgeführt, die nur mit exklusivem Datenbankzugriff erfolgen können. Darunter fallen zum Beispiel die Überprüfung der Datensatz-IDs. Zum anderen werden Defekte, die durch die normale Diagnose und die erweiterte Diagnose gefunden wurden,

## Kontakt

korrigiert und die interne Baumstruktur an den betreffenden Stellen neu aufgebaut.

**Vor dem manuellen Start eines Recover-Laufs sollte zunächst eine Diagnose ohne Recover-Option erfolgen, um Art und Umfang von eventuellen Defekten in der Datenbank zu ermitteln. Die Dauer des Recovers ist von der Anzahl der Datenbankdefekte, der Datenbankgröße, der Größe des Datenbankpuffers und der Leistungsfähigkeit der Hardware abhängig.**

- **Diagnoseoption "Schlüsselanalyse und -reparatur"**

Analog zur Diagnose mit Recover darf sich bei der Schlüsselanalyse und -reparatur kein Benutzer mehr in der Datenbank befinden und der Backup-Modus darf nicht aktiv sein. Die Schlüsselanalyse überprüft die Konsistenz von Datensätzen sowie die Konsistenz von den dazugehörigen Schlüsselwerten. Abweichungen werden automatisch korrigiert. Werden Abweichungen in Strukturen festgestellt, die zuerst durch ein Recover repariert werden müssten, wird keine Analyse des betreffenden Bereichs vorgenommen.

Die Dauer dieser Analyse ist sehr stark abhängig von der Anzahl der Datensätze in der Datenbank, dem eingestellten Datenbankpuffer und der Leistungsfähigkeit der Hardware. Die Analyse kann durchaus mehrere Stunden oder Tage in Anspruch nehmen. Um den Betrieb der Datenbank nicht unnötig lange zu unterbrechen, ist es daher unter Umständen sinnvoll, zunächst die Schlüsselanalyse auf einer Kopie der Datenbank durchzuführen. Je nach Ergebnis können beispielsweise Schlüsseldifferenzen bei wenigen Schlüsseln schneller durch eine manuelle Schlüsselreorganisation beseitigt werden, als wenn eine komplette Analyse auf der Originaldatenbank durchgeführt wird.



Eine vorherige Optimierung der Datenbank kann unter Umständen zu einer schnelleren Analyse beitragen.

### Starten der Diagnose

Eine Datenbankdiagnose kann auf verschiedene Arten gestartet werden. Die einfachste Art eine Diagnose zu starten ist es, diese über einen CONZEPT 16-Client aufzurufen. Dabei werden die Diagnoseresultate an den Client zurückgeliefert und angezeigt.

Über das CONZEPT 16-Script-Utility kann mit dem Kommando diag eine Diagnose gestartet werden. Ebenso ist es möglich eine Diagnose vom Server automatisch durchführen zu lassen. Der Datenbankserver kann so konfiguriert werden, dass täglich zu einer definierten Uhrzeit eine Diagnose einer Datenbank durchgeführt wird (siehe Konfiguration der Datenbanken). Nach einem Rollback (siehe Transaktionen / Rollback) wird automatisch eine Diagnose mit Recover durchgeführt. Ist beim Server die automatische E-Mail-Benachrichtigung eingestellt, werden Diagnoseresultate mit Fehlern automatisch per E-Mail versendet.

Das Diagnoseresultat wird unabhängig von der Art des Aufrufs im Pfad der Datenbank unter dem Datenbanknamen mit der Erweiterung .dgn gespeichert. In dieses kann dann zum Beispiel schon während des Recover-Laufs eingesehen werden, um den

## Kontakt

Umfang der Reparaturmaßnahmen zu ermitteln. Informationen zu den möglichen Einträgen im Diagnoseresultat stehen unter [Diagnoseresultate](#).

-  Die Diagnose wird auch bei geschlossener Datenbank gestartet. Dazu wird die Datenbank geöffnet, die Diagnose durchgeführt und die Datenbank wieder geschlossen.
-  Wird in einem Hot-Standby System eine Diagnose mit "Recover" oder "Schlüsselanalyse und -reparatur" auf der Primärdatenbank durchgeführt, trennt der CONZEPT 16-Server für die Dauer der Diagnose die Verbindung zum Sekundärsystem. Nach Diagnoseabschluss erfolgt eine Synchronisation der Sekundärdatenbank. Bei Datenbanken im Standby-Betrieb wird grundsätzlich keine automatische Diagnose durchgeführt.

## Kontakt

### CONZEPT 16-Server - Datenbankoptimierung

#### Beschreibung der Optimierung von CONZEPT 16-Datenbanken

Der CONZEPT 16-Server verfügt über eine Optimierungsfunktion. Diese Funktion organisiert die Segmente der Datenbank neu, wodurch eine bessere Performanz und ein geringerer Speicherverbrauch auf dem Speichermedium erreicht werden kann.



Vor der Durchführung einer Optimierung muss eine Sicherung der Datenbank erfolgen.



Bei der Optimierung wird der Cache der Datenbank geleert. D. h. der Inhalt aller temporären Dateien wird gelöscht. Zudem werden alle Selektionsmengen geleert. Vor der Verwendung einer Selektionsmenge muss diese wieder gefüllt werden (zum Beispiel mit SelRun()).

Bei der Optimierung wird zunächst eine Diagnose der Datenbankstruktur durchgeführt. Findet die Diagnose ein Problem in der Datenbank, erfolgt keine Optimierung. Bei der Durchführung sind 10 KB Hauptspeicher pro MB Datenbank erforderlich. Bei einer Datenbank von 10 GB werden also temporär 100 MB Hauptspeicher benötigt.

#### • Optimierung

Bei der Optimierung werden alle Segmente der Datenbank so angeordnet, dass keine leeren Segmente zwischen gefüllten Segementen vorhanden sind. Häufig zusammen benötigte Segmente werden hintereinander in die Datenbank geschrieben. Alle leeren Segmente befinden sich anschließend am Ende der Datenbank. Sollten viele leere Segmente vorhanden sein, werden diese aus der Datenbank entfernt. Die Optimierung ist die einzige Möglichkeit, die Datenbank zu verkleinern. Anschließend wird zur Sicherheit eine wiederholte Diagnose der Datenbankstruktur durchgeführt.

#### • Erweiterte Optimierung

Bei einer erweiterten Optimierung werden alle Baumstrukturen neu generiert, wobei eine maximale Datendichte in den Datensegmenten erzielt wird. Die Datenbank wird dadurch nochmals kleiner als bei der normalen Optimierung. Danach erfolgt zur Sicherheit eine wiederholte Diagnose der Datenbankstruktur. Zum Schluß wird die Standardoptimierung vorgenommen, die eine Neusortierung aller Datensegmente vornimmt.

Bei der erweiterten Optimierung werden auch alle nicht zuzuordnenden Baumstrukturen (in der Diagnose werden diese mit ??? angezeigt) gelöscht. Daneben wird bei der erweiterten Optimierung auch eine erweiterte Diagnose durchgeführt.

Nach einer Optimierung sind alle Selektionsmengen geleert. Bestehende Selektionsmengen müssen somit vor einem Zugriff auf die enthaltenen Datensätze mit den Befehlen SelRun() oder SelRecInsert() gefüllt werden.



Eine Optimierung sollte nicht durchgeführt werden, wenn das zugrundeliegende Plattensystem selbst eine Optimierung anhand der Zugriffshäufigkeiten auf die Festplattensegmente vornimmt.

## Kontakt

### CONZEPT 16-Server - Limitationen

#### Aufstellung der Limitationen von CONZEPT 16-Datenbanken

Folgende technische Limitationen sind beim Einsatz von CONZEPT 16-Datenbanken zu berücksichtigen:

Anzahl Benutzer* pro <u>Datenbank</u>	1000 (zusätzlich beschränkt durch die <u>Serverlizenzen</u> )
Datenbankgröße	8 TiB
Anzahl Datenräume pro Datenbank	8
<u>Dateien</u> pro Datenbank	999
Teildatensätze pro Datenbank	10.000
Teildatensätze pro Datei	10.000
Teildatensätze pro Seite Felder pro	254
Datenbank Felder pro Datei	65.000
	65.000
Felder pro Teildatensatz	200
Schlüssel pro Datenbank	16.000
Schlüssel pro Datei	255
Schlüsselfelder pro Datenbank	65.000
Schlüsselfelder pro Datei	65.000
Schlüsselfelder pro Schlüssel	32
Schlüsselgröße	950 Byte
Verknüpfungen pro Datenbank	16.000
Verknüpfungen pro Datei	255
Datensätze pro Datenbank	über 4000 Milliarden
Datensätze pro Datei Größe pro Datensatz	über 4 Milliarden 512 KiB
Anzahl der internen Texte	2 Milliarden
Größe eines internen Textes	128 MiB
Größe des P-Codes einer Prozedur 4	MiB

\* Die Anzahl der Benutzer wird durch die Lizenz beschränkt. Ist eine Lizenz auf 10 Benutzer beschränkt, können sich Benutzer von maximal 10 Rechnern gleichzeitig beim CONZEPT 16-Server anmelden (auch an unterschiedliche Datenbanken). Die Anzahl der Verbindungen zum CONZEPT 16-Server ist unabhängig von der Anzahl der lizenzierten Benutzer auf 1000 Verbindungen begrenzt. Wird ein neuer Benutzerkontext angelegt, unter anderem durch die Ausführung einer Prozedur oder Selektion beim Server, entspricht das einer weiteren Verbindung, die aber keinen lizenzierten Benutzer benötigt.

## Kontakt

### CONZEPT 16 - Installation des Servers

#### Installationsbeschreibung des CONZEPT 16-Servers

Der CONZEPT 16-Server besteht aus den Programmdateien des Datenbankservers, einer Lizenzdatei (c16.lic) und einem Lizenzdongle bzw. einer Lizenzidentitäts-Datei (c16\_xxxxxx.idn). Im Lizenzdongle bzw. in der Lizenzidentitäts-Datei wird die Lizenznummer des Datenbankservers festgehalten. Er stellt im eigentlichen Sinne die Lizenz des CONZEPT 16-Servers dar. In der Lizenzdatei wird die Lizenznummer des Dongles einer Server-Edition zugeordnet. Durch den Austausch der Lizenzdatei kann der Server also erweitert werden, ohne dass der Lizenzdongle ausgetauscht werden muss. Dies kann auch im laufenden Betrieb des Servers vorgenommen werden.

Bevor der Lizenzdongle auf das System aufgesteckt wird, müssen die mitgelieferten Dongle-Treiber installiert werden. Die betriebssystemeigenen Treiber sind für den Lizenzdongle nicht geeignet. Die Installation des Dongle-Treibers ist nicht notwendig, wenn eine Internetbasierte Lizenz verwendet wird. Vor der Installation des CONZEPT 16-Servers sollten zusätzlich die Systemvoraussetzungen zum Betrieb des Servers überprüft werden, damit die korrekte Server-Edition auf dem System eingerichtet werden kann.

Neben dem Verzeichnis für den Server, wird auch ein Verzeichnis für gemeinsam genutzte Ressourcen angelegt Resource. Die in diesem Verzeichnis liegenden Dateien werden vom Server, dem Control-Center und dem Log-Viewer verwendet.

Die nachfolgenden Kapitel gliedern sich folgendermaßen auf:

- Hardware-Empfehlung für den Datenbank-Server
- Lizenzmanagement
- Lizenztypen
- Lizenzinformationen
- Installation des Servers unter Windows
- Starten und Stoppen unter Windows
- Installation des Servers unter Linux
- Starten und Stoppen unter Linux

## Kontakt

### Hardware-Empfehlung für den Datenbank-Server

#### Aufstellung der empfohlenen Hardwarekonfiguration beim Einsatz des CONZEPT

16-Servers

Siehe [Blog](#)

Die Hardware des Datenbank-Servers beeinflusst wesentlich die Leistungsmerkmale des gesamten Datenbanksystems. Daher ist beim Einsatz einer Datenbank ein starkes Augenmerk auf die zu verwendende Hardware zu legen. Vor der Investition in die einzelnen Server-Bausteine muss jedoch der Einsatzzweck des Systems genau bestimmt werden. Dabei spielt die Datenbankapplikation die größte Rolle. Denn diese bestimmt die spätere Arbeitsweise mit der gesamten Software-Umgebung und bedingt damit die Hardware-Konfiguration des Datenbank-Servers.

Die Server-Hardware muss je nach Applikation bestimmte Voraussetzungen für einen performanten Einsatz erfüllen. Bei jeder Applikation und Installation müssen diese Voraussetzungen überprüft und gegebenenfalls variiert werden. Daher verstehen sich alle Angaben auch nur als ungefähre Richtwerte. Nachfolgend die Faktoren, die die Performanz eines Datenbanksystems beeinflussen:

- Die Anzahl der Anwender, die mit dem System arbeiten und
  - die Häufigkeit der Datenbankanfragen
  - Das zu erwartende Transaktionsvolumen, das die Anwendung erzeugt
  - Die voraussichtliche Größe der Datenbank, in Bezug auf die Laufzeit des Systems

### Anwenderzahl und Anfragehäufigkeit

Die Anzahl der Benutzer und die Häufigkeit der Anfragen wirkt sich in erster Linie auf die Prozessorbelastung aus. Zu empfehlen sind dabei unabhängig von der Benutzeranzahl mindestens zwei Kerne, sodass zwei logische Einheiten die Anfragen parallel bearbeiten können. Ab einer Benutzeranzahl von 50 Anwendern ist zu einem System mit 4 Prozessorkernen zu raten. Die Anzahl der Prozessorkerne lässt sich entsprechend der folgenden Tabelle weiterführen:

bis 50 Benutzer      2 - 4 Kerne

bis 100 Benutzer 4 - 8 Kerne

bis 200 Benutzer 8 oder mehr Kerne

Zu dem Aspekt der Benutzeranzahl muss natürlich auch die Häufigkeit der Anfragen entsprechend berücksichtigt werden. Bei einer hohen Anfragefrequenz sollte auch ein höherer Wert auf die Prozessorkonfiguration gelegt werden. Eine umfassende Selektionsverarbeitung ist beispielsweise ein Merkmal, das sich auch auf die Wahl des Prozessors auswirken muss.



Höhere Taktraten des Prozessors sind wichtiger als eine große Anzahl von Cores. Die vermutete Anfragehäufigkeit muss sich auch in der Hauptspeicherausstattung des Systems wiederspiegeln. Denn auch wenn der Server eine große Menge an Anfragen durch die Prozessoren verarbeiten kann, müssen dennoch die Daten aus der Datenbank gelesen werden. Eine gute Hauptspeicherausstattung führt ebenfalls zu einer schnelleren Verarbeitungszeit der Anfragen, da die Daten bereits im Cache

## Kontakt

vorgehalten werden können (siehe hierzu auch Transaktionsvolumen).

### Transaktionsvolumen

Der Einsatzzweck der Datenbankapplikation, die Applikationslogik und die Häufigkeit sowie das Volumen der Datenbankanfragen bestimmen das zu erwartende Transaktionsvolumen (siehe auch Transaktionsmanagement des CONZEPT 16 Servers). Um Transaktionen verarbeiten zu können, muss der Datenbank-Cache entsprechend eingerichtet werden. Bezuglich der Hardware-Konfiguration des Datenbank-Servers muss sich das Volumen der Transaktionen also direkt auf den zur Verfügung stehenden Hauptspeicher auswirken.

Der einer Datenbank zur Verfügung stehende Hauptspeicher muss in der Regel 10% der Datenbankgröße betragen. Mehr als diese 10% sind insbesondere bei vermehrtem Datenaufkommen ratsam, insofern es die Systemumgebung zulässt. Daher wird empfohlen den Datenbank-Server auf einer 64-Bit-Platform einzusetzen. Dazu eignen sich in erster Linie "Windows Server 2012" und der "Linux Kernel 3.x". Dort besteht dann die Möglichkeit den notwendigen Hauptspeicher für die Datenbanken einzurichten (siehe auch Architektur des Servers). Bei Datenbanken, die nur als Container für binäre Daten dienen, ist es nicht notwendig einen großen Cache anzulegen. Ein großer Datenbank-Cache ist nur nötig, wenn Daten überwiegend in der Baumstruktur einer CONZEPT 16-Datenbank organisiert werden (siehe Architektur der Datenbank).



Der Hauptspeicher eines Datenbank-Servers stellt nicht nur eine Verbindung zwischen Datenverarbeitung und Datenspeicherung her. Durch das Zwischenspeichern von gelesenen oder noch zu schreibenden Daten steuert er auch einen maßgeblichen Teil zur Performance des gesamten Systems bei. Daher muss dieser Komponente eine erhöhte Aufmerksamkeit geschenkt werden. In der Regel ist der Hauptspeicher die kritische Komponente eines Systems und führt gegebenenfalls zu einer höheren oder niedrigeren Performance.

Neben der Größe des Hauptspeichers werden auch die benötigten Input / Output-Kapazitäten des Storage-Systems vom Transaktionsvolumen beeinflusst. Selbst ein großer Hauptspeicher kann in der Regel nicht alle Informationen vorhalten. Die notwendigen Schreib- und Leseoperationen auf der Speichereinheit dürfen dann nicht zu einer inakzeptablen Verzögerung führen (siehe hierzu auch Datenbankgröße).

### Datenbankgröße

Die Größe der Datenbank beeinflusst nicht nur wie erwähnt die Größe des notwendigen Hauptspeichers, sondern ebenfalls wie das Transaktionsvolumen auch das notwendige Storage-System. Für das Arbeiten mit einem Datenbanksystem werden generell SSDs mit SATA- (Serial Advanced Technology Attachment) oder SAS- (Serial Attached SCSI) Anschluss mit einer möglichst niedrigen Latenz empfohlen.

Die Speichergrundlage eines Datenbanksystems sollte ein RAID-Verbund sein. Für eine optimierte Schreib- und Leseleistung empfiehlt sich der Einsatz eines Stripings (RAID 0). Dabei werden die Daten auf mehrere SSDs verteilt geschrieben, wodurch sich die einzelnen Zugriffsgeschwindigkeiten addieren. Um einem Fehlerfall im Storagebereich vorzubeugen, wird dies noch mit einem Mirroring (RAID 1)

## **Kontakt**

**kombiniert. Der Einsatz eines RAID 10 Systems eignet sich also am besten für den Einsatz eines Datenbanksystems.**

## Kontakt

### CONZEPT 16-Server - Lizenzmanagement

#### Management der CONZEPT 16-Serverlizenzen

Eine Lizenz des CONZEPT 16-Servers besteht aus der Lizenzidentität und dem Lizenzumfang. Die Lizenzidentität wird in Form einer sechsstelligen Lizenznummer in folgenden Varianten bereitgestellt:

- Lizenz mit Hardwareschutz (Dongle)
- Lizenz mit Softwareschutz (Internetbasierte Lizenz)
- Evaluierungslizenz

Der Lizenzumfang (Edition, Plattformen, maximale Anzahl der Datenbankbenutzer, Versionsstand usw.) wird unter Bezug auf die jeweilige Lizenznummer in einer separaten verschlüsselten Datei mit dem Namen c16.lic bestimmt (siehe auch Lizenztypen). Bei Änderungen des Lizenzumfangs stellt vectorsoft eine neue Version dieser Lizenzdatei bereit. Es muss dann lediglich die alte Lizenzdatei (c16.lic) durch die neue Lizenzdatei ersetzt werden (siehe Speicherorte von Konfigurationsdateien).

Der CONZEPT 16-Server muss zum Einlesen der neuen Lizenzinformationen nicht neu gestartet werden. Die Lizenzdatei muss nur dann ausgetauscht werden, wenn eine Änderung an der Lizenz erfolgt ist, oder die Versionsnummer des Programmstandes in der zweiten Stelle (zum Beispiel von Version 5.3.10 -> 5.4.00) wechselt. Der Austausch der Lizenzdatei kann automatisiert erfolgen (siehe Konfiguration des Servers).

Bei der Verwendung der Hot-Standby-Option werden zwei Datenbankserver gestartet, wobei aber nur einer der Server eine Lizenz besitzt. Die Lizenzierung auf dem Server ohne Dongle ist im Abschnitt Installation der Hot-Standby Option erläutert.

#### Einzelplatzlizenzen

Eine Einzelplatzlizenz (Lizenz mit nur einem Benutzer) kann entweder hardwaremäßig mit Dongle oder softwaremäßig mit einer verschlüsselten Identitätsdatei bereitgestellt werden. Der CONZEPT 16-Server kann mit dieser Lizenz nicht über das Netzwerk angesprochen werden. Daher muss bei der Serveradresse auch ein \* oder die Loopback-Adresse angegeben werden (127.0.0.1 oder localhost). Über die IP-Adresse des Netzwerks ist der Datenbankserver nicht erreichbar. Werden die Clients lokal auf dem System gestartet, können sich beliebig viele Clients mit dem Server verbinden.

#### Betrieb ohne Lizenz

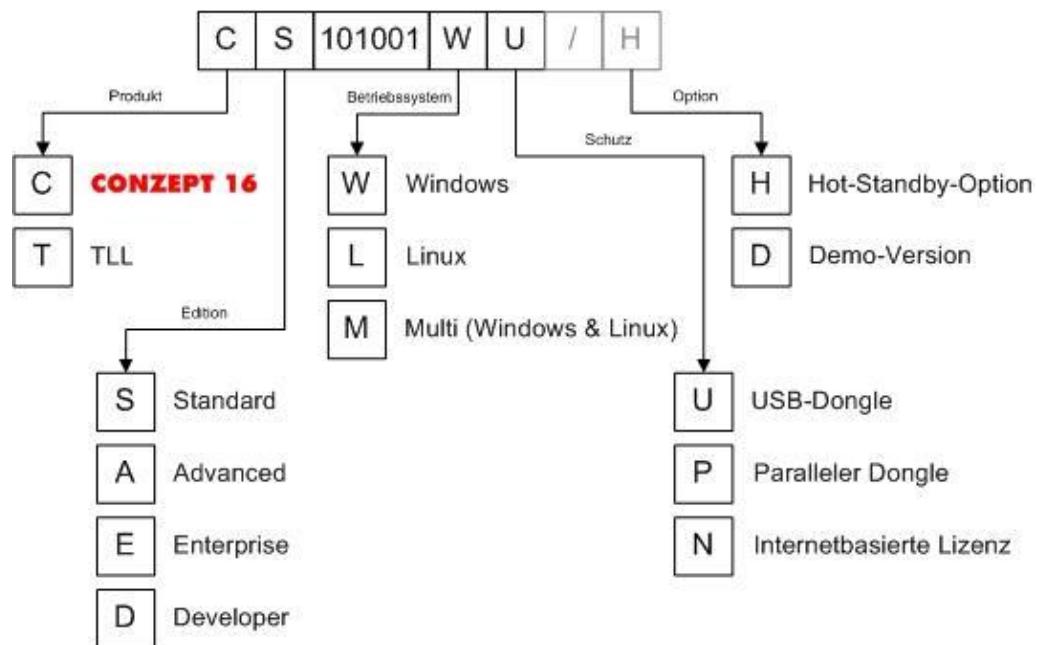
Der CONZEPT 16-Server kann auch ohne eine eingerichtete Lizenz gestartet werden. Der Manager-Prozess trägt dann entsprechende Fehlermeldungen in seine Log-Datei (c16\_serv\_mgr.lgb, siehe Log-Einträge) ein. Eine Datenbank kann dann allerdings nicht geöffnet werden.

## Kontakt

### Lizenztypen

#### Beschreibung der Lizenztypen des CONZEPT 16-Servers

Alle Lizenzen sind mit einer eindeutigen Lizenznummer versehen, aus der sich der Lizenztyp und die Art des Lizenzschutzes ersehen lässt. Die Lizenznummern sind wie folgt aufgebaut:



#### Das erste Zeichen - Das Lizenzprodukt

Es werden zwei Lizenztypen ausgeliefert. "C" bezeichnet eine unbefristete CONZEPT 16-Lizenz. Diese Lizenz wird nur mit einem Dongle (USB oder Parallelport) ausgeliefert. Mit "T" ist eine TLL-Lizenz (Time-limited License) gekennzeichnet. Die Lizenz besitzt keinen Dongle, der Lizenzschutz erfolgt über einen Lizenzserver.

#### Das zweite Zeichen - Die Edition

Es gibt vier verschiedene Lizenz-Editionen. Die Editionen werden durch das zweite Zeichen unterschieden. Je nach Lizenz-Edition stehen dem Server gewisse Funktionen zur Verfügung:

#### Übersicht über die Leistungsmerkmale der verschiedenen Lizenztypen

	Developer-Edition	Standard-Edition	Advanced-Edition	Enterprise
(D)		(S)	(A)	(E)
Server für Windows				
Server für Linux				
Server Multilizenz			X	
Server für 32-Bit				
Server für 64-Bit		1		1
1 bis 8 logische Prozessoren				

## Kontakt

9 bis 16 logische Prozessoren	X	
mehr als 16 logische Prozessoren	X	X
Replikationssätze erzeugen	X	
Windows-Standard-Client		
Windows-Advanced-Client	X	
PDF-Verarbeitung		
Windows-Druckertreiber	X	
DLL-Schnittstelle	X	
ODBC-Schnittstelle	X	X
Web-Schnittstelle	X	
SOA-Service	X	
PHP-Schnittstelle	X	X
Laufwerkstreiber	X	X

<sup>1</sup> Die Standard- und Advanced-Edition können auch auf 64-Bit-Plattformen eingesetzt werden. Der maximal zuweisbare Datenbankcache liegt dann allerdings bei 1 GB.

### Das dritte bis achte Zeichen - Die eindeutige Lizenznummer

Hierbei handelt es sich um eine eindeutige, sechsstellige Lizenznummer der CONZEPT 16-Serverlizenzen.

### Das neunte Zeichen - Das Betriebssystem

In diesem Zeichen wird das Betriebssystem kodiert, unter dem die Lizenz eingesetzt werden kann. CONZEPT 16-Serverlizenzen können entweder unter Windows, unter Linux oder unter beiden Betriebssystemen eingesetzt werden.

W Die Lizenz kann nur unter Windows-Betriebssystemen eingesetzt werden.

L Die Lizenz kann nur unter Linux eingesetzt werden.

M Die Lizenz kann unter Windows und Linux eingesetzt werden.

Befindet sich eine Evaluierungslizenz im Einsatz, wird das durch die Buchstaben EV im neunten und zehnten Zeichen angezeigt. Die vorangegangenen Zeichen entsprechen der Lizenznummer der Lizenz, aus der die Evaluierungsversion erzeugt wurde. Informationen zum Erstellen einer Evaluierungsversion befinden sich im Abschnitt Evaluierungslizenz.

### Das zehnte Zeichen - Der Lizenzschutz

CONZEPT 16-Lizenzen können mit einem Lizenz mit Hardwareschutz (Dongle) (USB oder Parallel-Port) oder einer Lizenz mit Softwareschutz (internetbasierte Lizenz) betrieben werden. Dieses Zeichen steht für den Typ des Lizenzschutzes.

U USB-Dongle

P Parallel-Dongle

## Kontakt

### N Internetbasierte Lizenzierung

#### Das Suffix - Optionen

Es werden folgende Lizenzoptionen angeboten:

#### H Hot-Standby D

##### Demo-Version

Die Hot-Standby-Option ist im Abschnitt CONZEPT 16-Server - Hot-Standby beschrieben.

Die Demo-Version ist eine zeitlich limitierte Lizenz, die bei einem potentiellen Neukunden installiert werden kann, um die Funktionsweise des Programms im Umfeld des Neukunden zu demonstrieren. Bis auf die zeitliche Limitierung entspricht sie der gleichen Lizenz ohne die Option "/D".

##### Evaluierungsversion

Die Lizenznummer einer Evaluierungsversion entspricht der Lizenznummer der Developer-Edition, mit der sie generiert wurde (siehe Evaluierungslizenz). Sie unterscheidet sich aber im neunten und zehnten Zeichen. Bei einer Evaluierungslizenz steht an diesen Stellen "EV". Die Lizenz "CD101432EV" ist also eine Evaluierungslizenz zum Beispiel von der Lizenz "CD101432MU/H". Da bei der Evaluierungslizenz kein Dongle benötigt wird und sie auch nur unter Windows installiert werden kann, ist auch eine Unterscheidung der Dongletypen und Betriebssysteme nicht notwendig.

## Kontakt

### Lizenz mit Hardwareschutz (Dongle)

#### Lizenzschatz über einen Dongle

Damit der Dongle gelesen werden kann, wird ein Dongle-Treiber benötigt, der vor dem Aufstecken des Dongles auf dem System installiert werden muss. Da der Dongle lediglich die Lizenznummer des CONZEPT 16-Servers enthält und der Umfang der Lizenz durch die Lizenzdatei bestimmt wird, ist eine Erweiterung des Lizenzumfangs ohne Austausch des Dongles möglich.

Kann eine gültige Lizenz im laufenden Betrieb nicht mehr gelesen werden (zum Beispiel durch Entfernen des Dongles), können trotzdem noch vierzehn Tage lang Datenbanken geöffnet werden, falls der Server zwischenzeitlich nicht neu gestartet wird. Wird das Problem bei noch laufendem Server beseitigt, wird die Lizenz beim Öffnen einer Datenbank, spätestens aber nach einer Minute wieder erkannt.

Wenn die Lizenz auf einem System korrekt eingerichtet ist, kann der Inhalt einer Lizenzdatei ausgelesen werden. Dazu muss das Programm c16\_licinfo.exe im Serververzeichnis gestartet werden. Hier können dann Informationen über die Lizenznummer, die Anzahl der zugelassenen Benutzer und die Lizenz-Edition angezeigt werden. Die Lizenz-Edition gibt Aufschluss darüber, welche Clients der Datenbankserver bedienen kann (siehe [Lizenztypen](#)). Die Angabe der zugelassenen Benutzer bezieht sich auf die maximal gleichzeitig am Server angemeldeten Datenbankanwender.

### Voraussetzungen

Dongles sind in zwei unterschiedlichen Varianten für den USB-Anschluss oder den Parallel-Port verfügbar. Damit einer dieser Dongletypen verwendet werden kann, muss eine entsprechende Schnittstelle vorhanden sein. Für Linux-Betriebssysteme werden nur USB-Dongles angeboten.

Um Fehlerquellen zu vermeiden, sollten keine Adapter oder USB-Hubs zum Anschluss des Dongles verwendet werden.

 Unter Windows-Betriebssystemen darf der USB-Dongle nicht vor der Installation des Dongle-Treibers auf die Schnittstelle gesteckt werden. Dem Gerät wird sonst ein generischer Treiber zugewiesen, über den eine Lizenzabfrage nicht möglich ist.

Zum Betrieb des Dongles (Parallel und USB) wird ein Dongle-Treiber benötigt. Unter Windows wird der Treiber automatisch installiert. Unter Linux wird er durch ein mitgeliefertes Skript installiert (siehe [Installation des Servers unter Linux](#)).

### Überprüfung des Lizenzschatzes

Zum Starten des CONZEPT 16-Servers muss der Dongle aufgesteckt sein. Während der Laufzeit des Servers wird der Dongle in regelmäßigen Abständen abgefragt. Wird dabei der Dongle nicht gefunden (zum Beispiel weil die Schnittstelle oder der Dongle defekt sind), wird der Server nicht sofort beendet. Er läuft bis zu 14 Tagen weiter. In der Protokolldatei des Managerprozesses wird das Fehlen des Dongles protokolliert (siehe [Log-Einträge](#)) und auch eine entsprechende Fehlermail versendet (falls der Server entsprechend konfiguriert ist: [Automatische E-Mail-Benachrichtigung](#)). In

## Kontakt

dieser Zeit kann der CONZEPT 16-Server nicht neu gestartet werden.

**Wurde der CONZEPT 16-Server ohne einen Lizenzschutz gestartet, kann keine Datenbank geöffnet werden. Der Server überprüft in kurzen Zeitabständen und wenn eine Datenbank geöffnet werden soll, ob ein Dongle vorhanden ist.**

**Wird der Dongle korrekt erkannt, befinden sich folgender Eintrag in der Protokolldatei des Managerprozesses:**

**Dongle read successfully**

### Lizenzdatei

**In der Lizenzdatei ist der Umfang der Lizenz beschrieben. Die Datei ist codiert und wird mit einem Schlüssel aus dem Dongle dekodiert. Sie enthält Informationen über den Lizenztyp, die Anzahl der Benutzer und welche Optionen vorhanden sind. Die Datei befindet sich im gleichen Verzeichnis, wie die Konfigurationsdateien des Servers (siehe Speicherorte von Konfigurationsdateien). Unter Linux-Betriebssystemen muss sich die Datei im gleichen Verzeichnis befinden, wie der Server. Die Datei wird vom Server nur gelesen. Es ist somit ausreichend, wenn Leserechte für diese Datei gegeben werden.**

**Bei Änderungen an der Lizenz muss nur die vorhandene Lizenzdatei durch eine neue Lizenzdatei ersetzt werden. Nach einigen Minuten verfügt dann der Server über die entsprechenden Änderungen.**

**Die Lizenzdatei kann frei kopiert werden. Sie ist damit unempfindlich gegenüber Sicherungen oder auch dem Wiederherstellen des Servers. Bei der Migration des Datenbankservers auf eine andere Hardware sollte eine Neuinstallation von CONZEPT 16 erfolgen, da dabei der Dongle-Treiber installiert und der CONZEPT 16-Server automatisch als Dienst eingetragen wird.**

## Kontakt

### Lizenz mit Softwareschutz (internetbasierte Lizenz)

#### Lizenzschatz über Lizenz-Server

Siehe Wechsel auf  
anderen  
Server

(Blog),  
Lauffähigkeit  
(Blog)

Die Lizenznummer wird in Form einer verschlüsselten Datei bereitgestellt (Identitätsdatei). Der Dateiname lautet c16\_xxxxxx.idn, wobei in den Platzhaltern x die jeweilige Lizenznummer steht. Der Lizenzumfang ergibt sich genauso wie bei der Dongle-Lizenz aus der Lizenzdatei c16.lic.

Identitätsdatei und Lizenzdatei werden von der Serversoftware benutzt und müssen sich im Datenverzeichnis des Datenbankservers befinden. Die Serversoftware benötigt Leserechte auf die Lizenzdatei und Schreib-/Leserechte auf die Identitätsdatei. Mit Hilfe des CONZEPT 16-Control-Centers können die beiden Dateien automatisch in das Verzeichnis kopiert werden. Während des Betriebs des Servers können sowohl die Identitätsdatei als auch die Lizenzdatei zu Sicherungszwecken gelesen werden, die Identitätsdatei kann jedoch nicht überschrieben werden.

Es darf sich nur eine einzige Identitätsdatei im Datenverzeichnis des Servers befinden.

### Netzwerkkommunikation

Das Lizenzverfahren setzt die Kommunikation des Datenbankservers mit vectorsoft-eigenen Service-Rechnern voraus, die nachfolgend als Lizenzserver bezeichnet werden. Die Kommunikation wird über das TCP/IP-basierte HTTP-Protokoll unter Verwendung einer Internet-Verbindung durchgeführt.

Zu diesem Zweck muss der CONZEPT 16-Server im Bedarfsfall (siehe Nutzbarkeit der Software) über das Internet eine TCP/IP-Verbindung mit einem der Lizenzserver auf Zielport 80 herstellen können, um eine Lizenzabfrage durchzuführen. Aktuell sind vier Lizenzserver mit den Hostnamen sina1.c16.net, sina2.c16.net, sina3.c16.net und sina4.c16.net vorhanden.

### Nutzbarkeit der Software

Die Identitätsdatei hat eine zeitlich beschränkte Gültigkeit, die durch regelmäßige Anfragen an einen Lizenzserver immer wieder verlängert wird. Die Nutzbarkeit der Serversoftware ist somit abhängig von der Durchführbarkeit von Lizenzabfragen. In den folgenden Fällen muss eine Lizenzabfrage unmittelbar beim Start des CONZEPT 16-Servers möglich sein, da ansonsten die Serversoftware nicht nutzbar ist:

1. Bei der erstmaligen Verwendung der Lizenz.
2. Bei einer Änderung der Rechnerkonfiguration des Datenbankservers.
3. Bei einem Wechsel der Serversoftware auf einen anderen Rechner.
4. Bei Verwendung einer früheren Version der Identitätsdatei.

## Kontakt

**Konnte sich der CONZEPT 16-Server mit einem Lizenzserver verbinden, stehen folgende Einträge in der Protokolldatei des Manager-Prozesses:**

```
Successful connect to License Server sinal.c16.net
```

**In allen anderen Fällen muss die Kommunikation zwischen Serversoftware und den Lizenzservern mindestens über einen zusammenhängenden Zeitraum von 15 Minuten innerhalb von 24 Stunden möglich sein, um die Durchführung einer Lizenzabfrage zu gewährleisten.**

**Konnte sich der CONZEPT 16-Server mit einem Lizenzserver verbinden, stehen folgende Einträge in der Protokolldatei des Manager-Prozesses:**

```
License Identity file is valid (150012) License is valid (CD150012MN/H, 100 user(s), release 5.5) S
```

Nach einer erfolgreichen Lizenzabfrage wird in der Identitätsdatei eine zusätzliche Gültigkeitsdauer im Bereich zwischen 8 und 15 Tagen eingetragen. Während dieser Gültigkeitsdauer ist die Nutzung der Serversoftware auch ohne erfolgreiche Lizenzabfragen möglich, beispielsweise bei einer Störung der Internetverbindung. Wenn bis zum Ablauf der Gültigkeit keine erfolgreiche Lizenzabfrage mehr durchgeführt werden kann, ist die Serversoftware nach dem Ablaufzeitpunkt nicht mehr nutzbar. Rechtzeitig vor Ablauf der Gültigkeit werden die Nutzer durch geeignete Fehlermeldungen auf diese Problematik hingewiesen.

### Übermittelte Daten in Lizenzabfragen

**Bei einer Lizenzabfrage werden die folgenden Daten an einen Lizenzserver gesendet:**

- Lizenznummer
- aktueller Lizenzschlüssel aus der Identitätsdatei
- Betriebssystemversion
- Menge des installierten Hauptspeichers
- Anzahl von logischen Prozessoren
- Hash-Wert des Rechnernamens

Die übermittelten Daten ermöglichen keine direkte Identifizierung des Lizenzanwenders. Anhand der übermittelten Lizenznummer kann jedoch eine Zuordnung der Lizenz zu einem bestimmten Anwender möglich sein.

Die Antwort des Lizenzservers enthält auch einen neuen Lizenzschlüssel, der in der Identitätsdatei gespeichert wird. Das Ergebnis der letzten Lizenzabfrage wird ebenfalls in der Identitätsdatei gespeichert und kann in den Lizenzinformationen von Lizenzen mit Softwareschutz eingesehen werden:

- Ungültige Lizenzabfrage (0)

Die Lizenzabfrage konnte nicht erfolgreich durchgeführt werden. Nähere Informationen stehen im Protokoll des Manager-Prozesses. Die Gültigkeit der Lizenz wurde nicht verlängert.

- Erste Inbetriebnahme (1)

## Kontakt

**Die Lizenz-Identitätsdatei wurde zum ersten Mal verwendet. Die Lizenzabfrage hat ein positives Ergebnis. Die Gültigkeit der Lizenz wurde eingetragen.**

- Änderung der Konfiguration (2)

**Die Konfiguration des Rechners, auf dem der CONZEPT 16-Server installiert ist, hat sich geändert, oder der CONZEPT 16-Server wurde auf einem anderen Rechner installiert (siehe Änderung der Rechnerkonfiguration).**

- Lizenzabfrage mit korrektem Schlüssel (3)

**Die Lizenzabfrage wurde mit der aktuellen Lizenz-Identitätsdatei durchgeführt, Änderungen am Rechner wurden nicht vorgenommen. Dies sollte der Normalfall sein.**

- Lizenzabfrage mit altem Schlüssel (4 und 5)

**Es wurde eine ältere Identitätsdatei verwendet. Dieses Ergebnis erfolgt, wenn eine ältere Identitätsdatei verwendet wird, aber auch bei einer Mehrfachnutzung der Lizenz.**

**Die Anfrage an den Lizenzserver und die darauf folgende Antwort, wird in der Datei c16\_serv\_mgr.http im Verzeichnis des Servers protokolliert. Die Datei kann mit einem beliebigen Text-Editor geöffnet und eingesehen werden.**

### Änderungen der Rechnerkonfiguration

**Jede der folgenden Änderungen stellt eine Konfigurationsänderung des Datenbankservers dar:**

- Änderung des verwendeten Betriebssystems (beispielsweise von Windows Server 2003 auf Windows Server 2008)
- Erhöhung oder Verringerung des installierten Hauptspeichers
- Änderung der Anzahl von logischen Prozessoren
- Änderung des Rechnernamens

**Nach einer Konfigurationsänderung muss beim Start der Serversoftware eine erfolgreiche Lizenzabfrage durchgeführt werden, da ansonsten die Serversoftware nicht verwendet werden kann.**

### Installation auf einem anderen System

**Die Installation und Inbetriebnahme der Serversoftware mit derselben Lizenz auf einem anderen Datenbankserver stellt eine Konfigurationsänderung dar. Es ist dabei zu beachten, dass zur Vermeidung einer Mehrfachnutzung die Serversoftware auf dem ursprünglichen Datenbankserver deaktiviert wird.**

### Verwendung von Datensicherungen

**Bei Verlust der aktuellen Identitätsdatei kann die Sicherung einer beliebigen älteren Version der Identitätsdatei verwendet werden. Nach der Wiederherstellung der Identitätsdatei muss beim Start der Serversoftware eine erfolgreiche Lizenzabfrage durchgeführt werden, da ansonsten die Serversoftware nicht verwendet werden kann.**

## Kontakt

### Kopierschutz

**Der Kopierschutzmechanismus des Lizenzverfahrens gestattet maximal zwölf Konfigurationsänderungen (siehe [Änderung der Rechnerkonfiguration](#)) oder Wiederherstellungen (siehe [Verwendung von Datensicherungen](#)) pro Jahr wobei zwischen zwei Konfigurationsänderungen oder Wiederherstellungen ein Mindestabstand von dreißig Tagen eingehalten werden muss.**

**Bei häufigeren Konfigurationsänderungen beziehungsweise Wiederherstellungen oder kürzeren Zeitabständen kann unter Umständen vorübergehend keine erfolgreiche Lizenzabfrage mehr durchgeführt werden, wodurch sich die Gültigkeitsdauer (siehe [Nutzbarkeit der Lizenz](#)) nicht mehr verlängert. Sollten aufgrund besonderer Umstände Konfigurationsänderungen oder Wiederherstellungen in kürzerem Abstand oder in größerer Zahl erforderlich sein, kann dies nach besonderer Abstimmung mit vectorsoft erfolgen.**

### Mehrfachnutzung

**Der Kopierschutzmechanismus erkennt aufgrund von Art und Zeitpunkt der Lizenzabfragen einen gleichzeitigen Betrieb der Lizenz auf mehreren Datenbankservern. In diesem Fall erhält nur der Datenbankserver eine Verlängerung der Gültigkeit, auf dem die Serversoftware am längsten in Betrieb ist. Alle anderen Datenbankserver erhalten keine Verlängerung der Gültigkeit mehr, wodurch die Serversoftware nach Ablauf des Gültigkeitszeitraums auf diesen Rechnern nicht mehr genutzt werden kann.**

### Sonstiges

**Wenn bei einer Lizenzabfrage eine Systemzeitdifferenz von mehr als einer Stunde zwischen Datenbankserver und Lizenzserver festgestellt wird, korrigiert die Serversoftware die Systemzeit. Falls die Systemzeit nicht korrigiert werden kann, akzeptiert die Serversoftware maximal eine Systemzeitdifferenz von 24 Stunden. Bei einer größeren Systemzeitdifferenz kann die Serversoftware nicht genutzt werden.**

## Kontakt

### Evaluierungslizenz

#### Zeitlich begrenzte Lizenz zur Evaluierung

Die Evaluierungslizenz ist ein eigener Lizenztyp. Die Besonderheit der Evaluierungslizenz besteht darin, dass sie ohne Dongle oder Identitätsdatei, dafür aber mit einer speziellen Evaluierungs-Lizenzdatei betrieben wird. Des Weiteren können mit der Evaluierungslizenz nur spezielle Evaluierungsdatenbanken geöffnet werden. In der Evaluierungslizenzdatei ist eine zeitliche Beschränkung festgehalten. Ist diese abgelaufen und wird die Lizenzdatei nicht erneuert, können auch keine Evaluierungsdatenbanken mehr geöffnet werden. In den Evaluierungsdatenbanken können zudem bis maximal 1000 Datensätze pro Datei angelegt und verwaltet werden.

Mit einer Evaluierungslizenz kann der CONZEPT 16-Server auch in einem Netzwerk betrieben werden. Es können sich bis zu fünf lizenzierte Clients gleichzeitig mit dem Server verbinden. Pro Rechner, der an beliebig vielen Datenbanken des Servers angemeldet ist, wird ein Benutzer lizenziert.

#### *Erstellen einer Evaluierungslizenz:*

Die Evaluierungslizenzdatei wird bei Abschluss oder Verlängerung eines Software-Pflegevertrages einer Developer-Edition zur Verfügung gestellt beziehungsweise erneuert. Sie enthält die Lizenznummer der Developer-Edition. Läuft der CONZEPT 16-Server mit einer Evaluierungslizenzdatei, können nur Datenbanken geöffnet werden, welche mit der entsprechenden Developer-Lizenz in eine Evaluierungsdatenbank umgewandelt wurden. Um eine Evaluierungsdatenbank zu erzeugen, muss die Datenbank mit dem CONZEPT 16-Advanced-Client der Developer-Edition geöffnet und mit dem Menüpunkt "Datei / In Evaluierungslizenz umwandeln" umgewandelt werden.



Dabei ist zu beachten, dass die umzuwandelnde Datenbank nicht auf eine bestimmte Lizenznummer reserviert sein darf, da sonst der Menüpunkt "Datei / In Evaluierungslizenz umwandeln" nicht erscheint!



Die Umwandlung in eine Evaluierungsdatenbank muss mit einer Kopie der Datenbank erfolgen. Nach der Umwandlung kann die Evaluierungsdatenbank nicht mehr mit einer normalen Lizenz geöffnet werden.

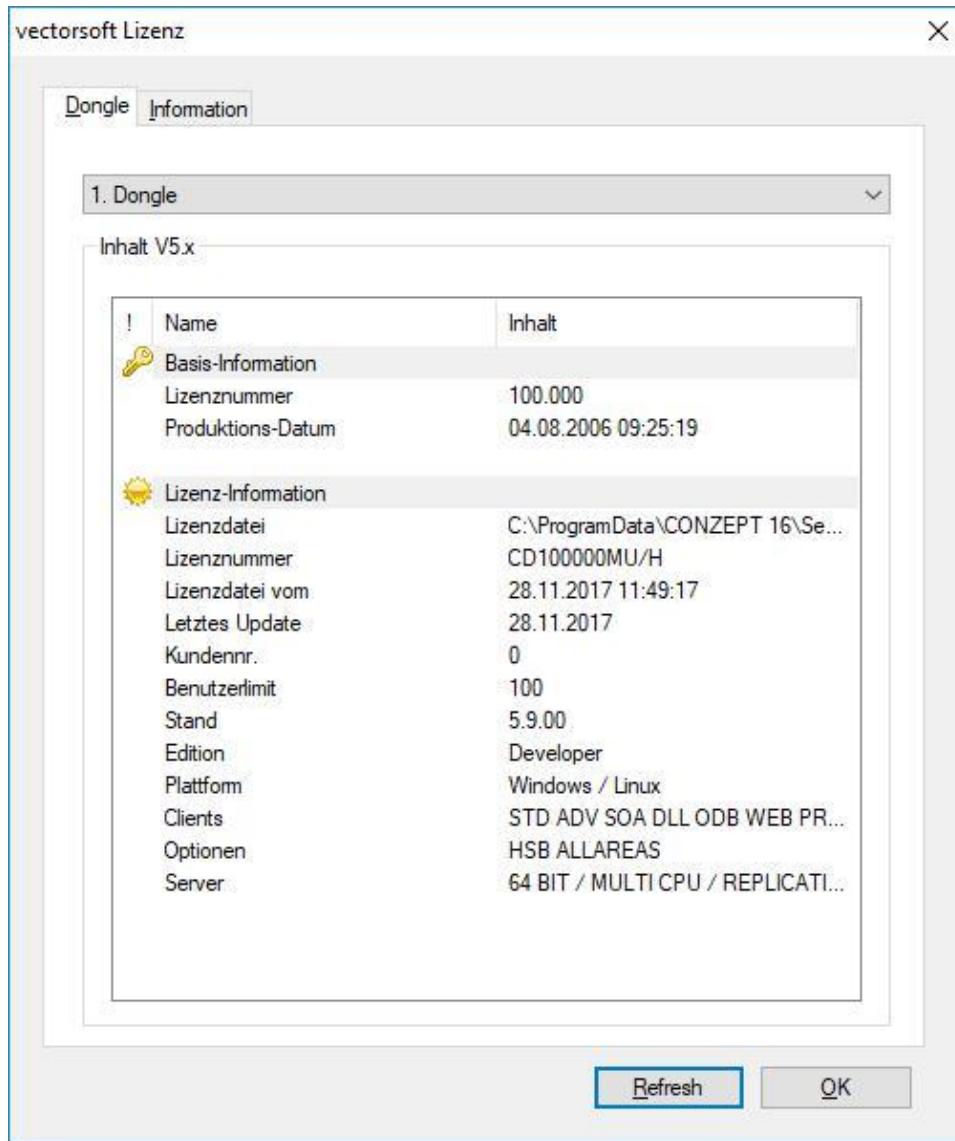
## Kontakt

### Lizenzinformationen

#### Informationen zu einer Lizenz ermitteln

Das Werkzeug c16\_licinfo.exe ist für das Auslesen der Lizenzdaten des Dongles bzw. der Lizenz-Identitätsdatei (\*.idn) und der Lizenzdatei (c16.lic) zuständig. Zum Auslesen der Lizenzinformation ist ein korrekt installierter CONZEPT 16 Lizenz-Dongle oder eine Lizenz-Identitätsdatei notwendig. Eine Ausnahme bildet die Evaluierungs Lizenz, bei der die Informationen auch ohne Lizenz angezeigt werden können. Die Lizenzdatei kann über die Kommandozeile als Parameter in Form des Dateipfades angegeben werden. Ansonsten wird diese bei den CONZEPT 16 Konfigurationsdateien oder anschließend im Startverzeichnis des Programms c16\_licinfo.exe gesucht.

Das Programm kann entweder aus dem Installationsverzeichnis des CONZEPT 16-Servers (zum Beispiel c:\Program Files\CONZEPT 16\Server) oder über das Control-Center gestartet werden.



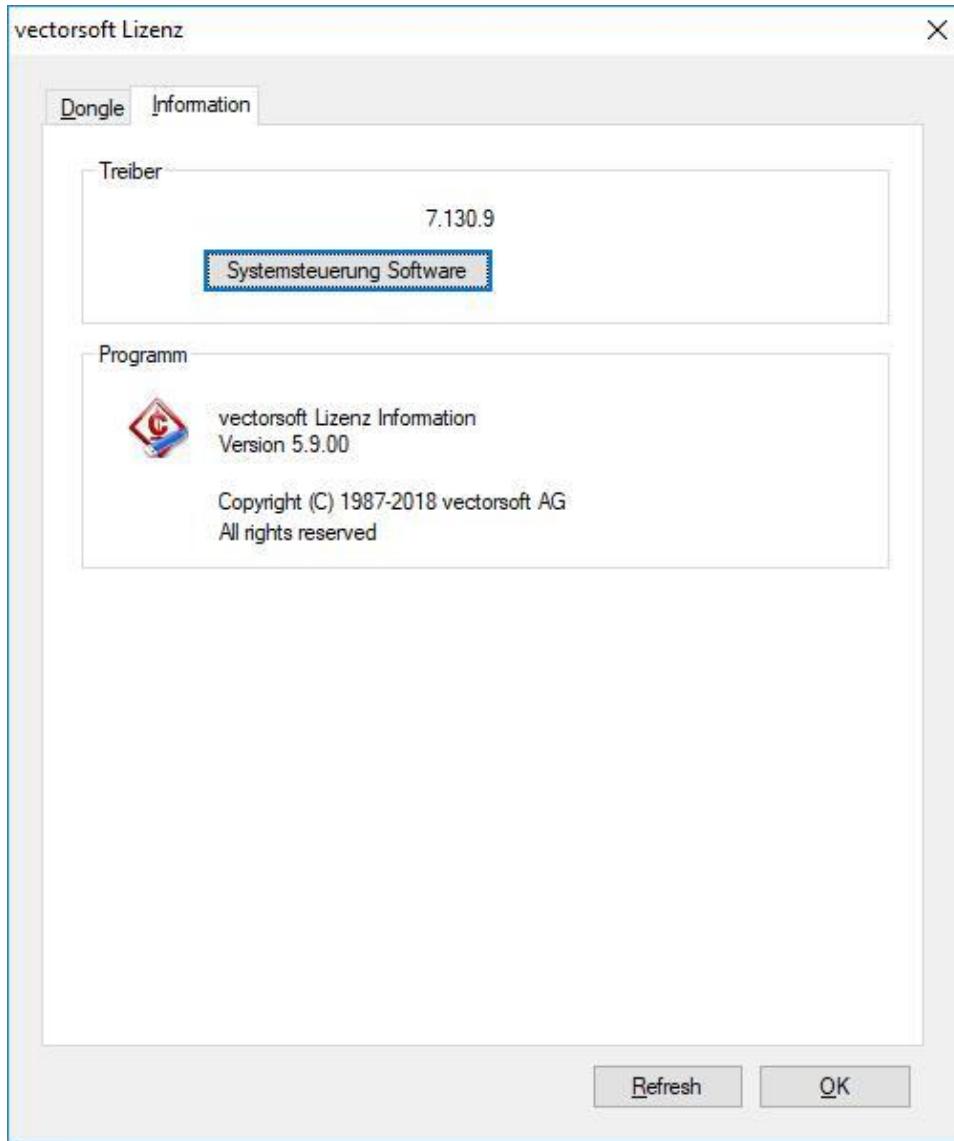
Abhängig von der verwendeten Lizenzierung werden verschiedene Informationen auf

## Kontakt

der Seite "Dongle" angezeigt. Die angezeigten Informationen sind in den Abschnitten Lizenzinformationen aus der Lizenzdatei, Lizenzinformationen von Lizenzen mit Softwareschutz und Lizenzinformationen von Lizenzen mit Hardwareschutz erläutert.

## Information

Die Seite "Information" beinhaltet Informationen über den Dongle-Treiber und das Programm selbst.



### • Treiber

Hier wird, falls installiert, die Version des Dongle-Treibers angezeigt.

Falls die Treiber installiert bzw. deinstalliert werden sollen, kann über den Button "Systemsteuerung Software" das Fenster "Software" der Systemsteuerung von Windows aufgerufen werden.

### • Programm

## **Kontakt**

**Hier werden einzelne Informationen des Programms angezeigt.**

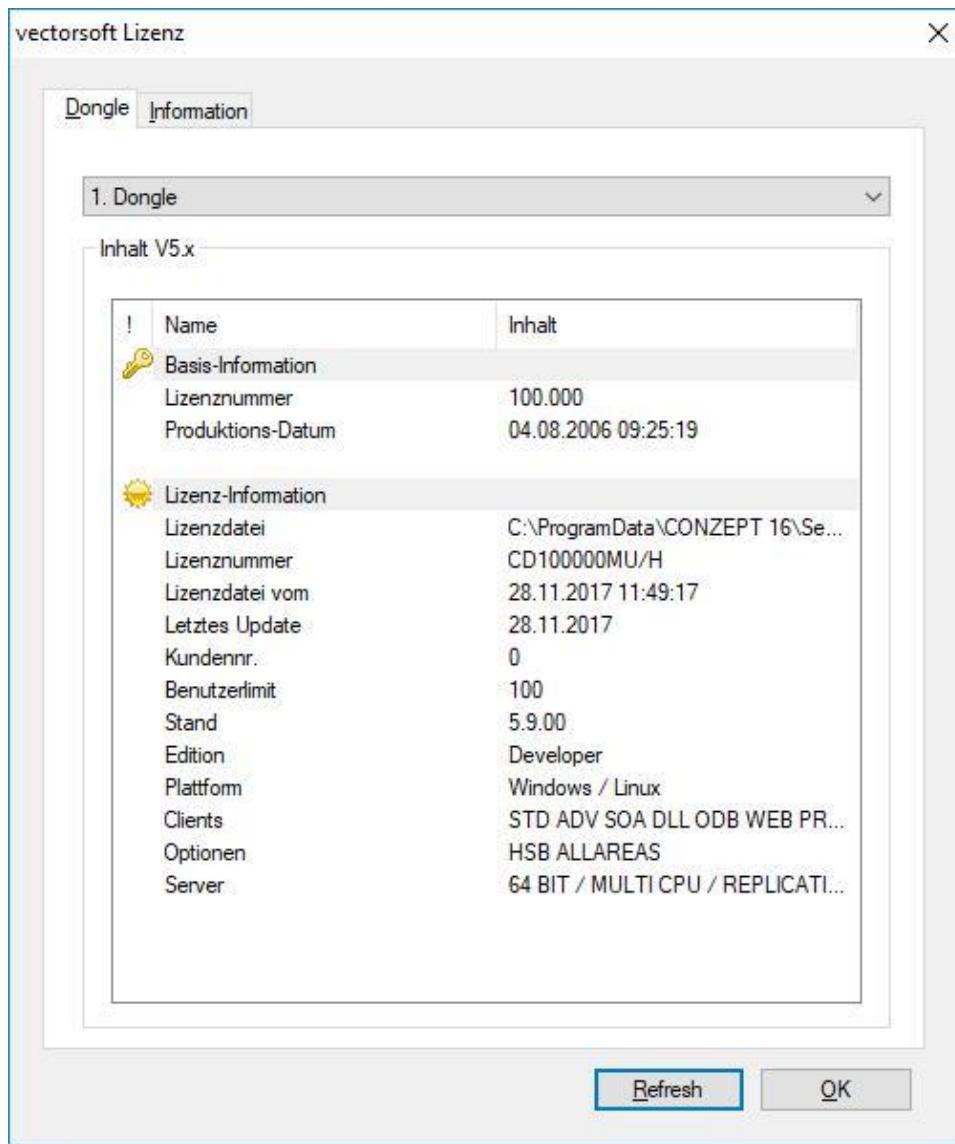
**Die Informationen zu einer Lizenz müssen nicht an dem Rechner erhoben werden, wo auch die Lizenz eingesetzt wird. Ist auf einem Rechner der Dongle-Treiber installiert, können zusammen mit der entsprechenden Lizenzdatei die Informationen abgefragt werden. Wird auf diese Maschine ein anderer Dongle gesteckt, können auch diese Informationen ausgelesen werden.**

## Kontakt

Lizenzinformationen aus der Lizenzdatei

Informationen über die Lizenzausprägung

Die folgenden Informationen werden in der Lizenzdatei (c16.lic) gespeichert.



- Lizenz-Information

- ◆ Lizenzdatei

- Hier wird der Speicherort der Lizenzdatei angezeigt.

- ◆ Lizenznummer

- Hier wird die Lizenznummer der Lizenzdatei angezeigt. Was die einzelnen Zeichen der Lizenznummer aussagen kann im Abschnitt Lizenztypen nachgelesen werden.

- ◆ Lizenzdatei vom

- Hier wird der Produktionszeitpunkt der Lizenzdatei angezeigt.

## Kontakt

### ♦ Letztes Update

Dies ist das Datum der letzten Änderung an der Lizenz.

### ♦ Ablaufdatum

Handelt es sich bei der Lizenz um eine zeitlich limitierte Lizenz, wird hier das Ablaufdatum der Lizenz wird hier angezeigt. Rechtzeitig vor Ablauf der Lizenz erfolgt ein Eintrag in die Protokolldatei. Ist die E-Mail-Benachrichtigung beim CONZEPT 16-Server eingetragen, wird ebenfalls eine E-Mail versenden.

### ♦ Kundennr.

Die zu der Lizenz zugehörige Kunden-Nummer wird in diesem Feld angezeigt.

### ♦ Benutzerlimit

Hier wird die unterstützte Anzahl der gleichzeitig angemeldeten Benutzer ausgegeben. Alle Clients einer Arbeitsstation zählt ab Clientversion 5.6 nur als ein Benutzer.

### ♦ Stand

In diesem Feld ist angegeben, welche Version von CONZEPT 16 mit dieser Lizenz maximal betrieben werden kann.

### ♦ Edition

Hier wird ausgegeben, welche Edition die Lizenz unterstützt. Dabei unterscheidet man zwischen den Editionen Standard, Advanced, Enterprise und Developer. Nähere Information und Unterschiede zu den Editionen befinden sich im Abschnitt Lizenztypen.

### ♦ Plattform

Hier werden die von der Lizenz unterstützten Betriebssysteme angezeigt. Erläuterung der Anzeige:

Windows      Die Lizenz kann nur auf einem Windows-System verwendet werden.

Linux      Die Lizenz kann nur auf einem Linux-System verwendet werden.

Windows /      Die Lizenz kann entweder auf einem Linux-System oder auf einem Windows-System verwendet werden.

### ♦ Clients

Hier wird angezeigt, welche Clients mit dieser Lizenz zur Verfügung stehen.

Erläuterung der Anzeige:

STD Standard-Client (c16\_winc.exe)

ADV Advanced-Client (c16\_apgi.exe)

## Kontakt

**DLL DLL-Schnittstelle (c16\_pgxw.dll)**

**ODB ODBC-Schnittstelle**

**WEB Web-Schnittstelle**

**PRT Druckertreiber-Unterstützung**

**PHP PHP-Schnittstelle**

### ♦ Optionen

Hier wird angezeigt, welche Optionen mit dieser Lizenz zur Verfügung stehen.

Erläuterung der Anzeige:

**HSB Hot-Standby-Option**

**PDF PDF-Option**

### ♦ Server

Hier wird angezeigt, auf welchen Systemen CONZEPT 16 mit dieser Lizenz betrieben werden kann.

Erläuterung der Anzeige:

**64-BIT** CONZEPT 16 kann auch auf Systemen mit 64-Bit-Architektur betrieben werden.

**CPU <= 8** CONZEPT 16 kann nur auf Systemen mit maximal acht logischen Prozessoren betrieben werden.

**CPU <= 16** CONZEPT 16 kann nur auf Systemen mit bis zu 16 logischen Prozessoren betrieben werden.

**MULTI CPU** CONZEPT 16 kann auf Systemen mit beliebiger Anzahl von Prozessoren betrieben werden.

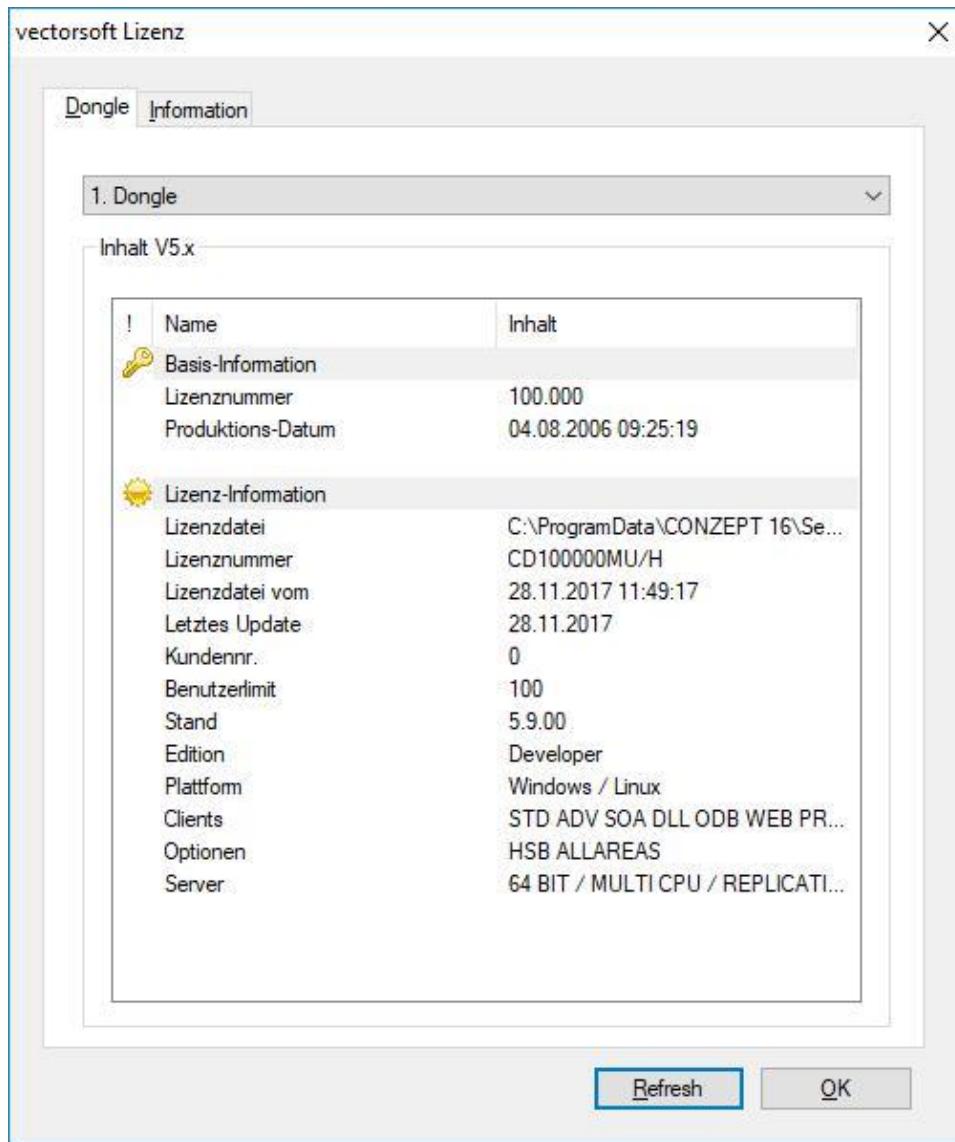
**REPLICATION** Der CONZEPT 16-Server kann Replikationsdateien erzeugen.

## Kontakt

Lizenzinformationen von Lizenzen mit Hardwareschutz Anzeige

der Informationen von Dongle-Lizenzen

Folgende Informationen werden bei einer Dongle-Lizenz im Dongle abgelegt. In einem USB- und einem Parallel-Port-Dongle werden die gleichen Informationen gespeichert.



- Basis-Information

- ◆ Lizenznummer

- Hier wird die Lizenznummer angezeigt.

- ◆ Produktions-Datum

- Hier wird angezeigt, wann die Lizenz produziert wurde.

Die Informationen im Bereich "Lizenz-Information" sind im Abschnitt

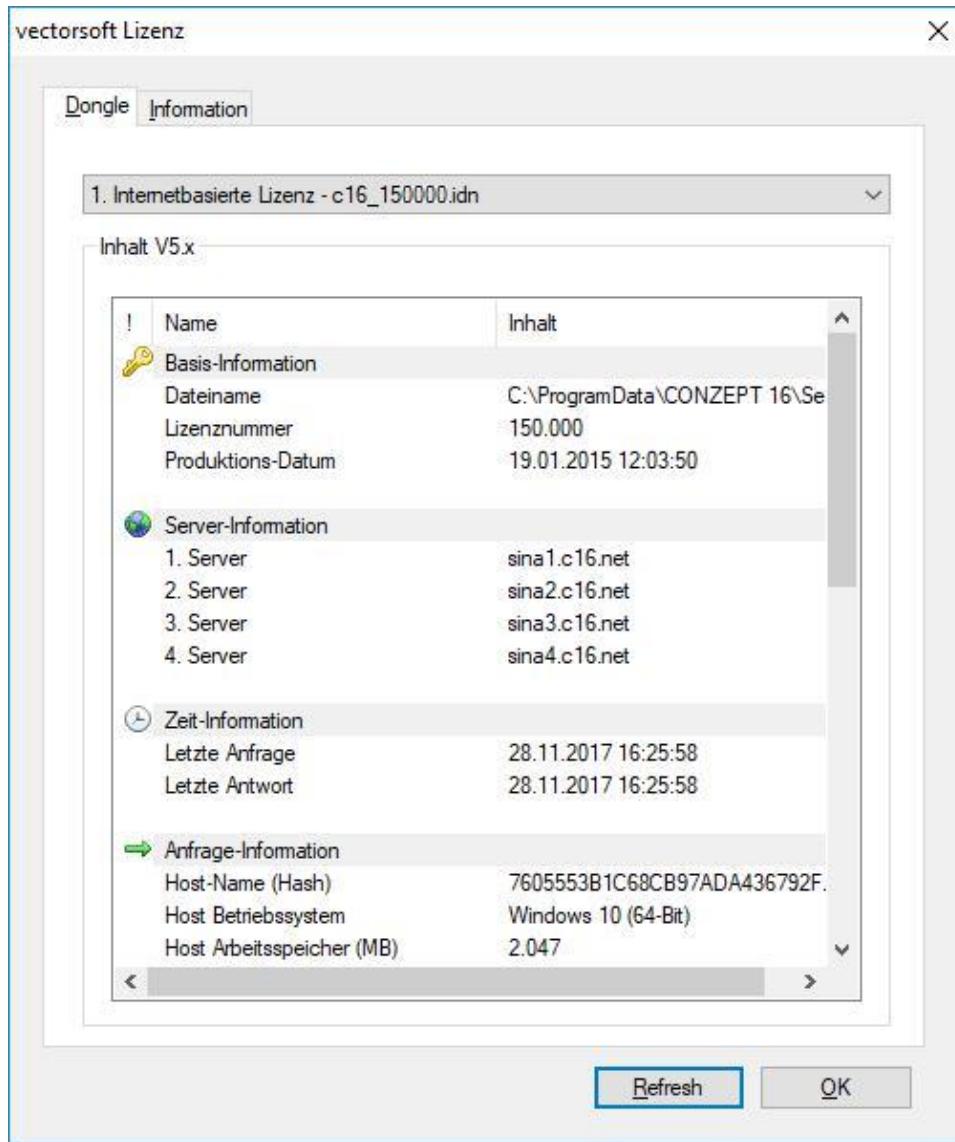
Lizenzinformationen aus der Lizenzdatei beschrieben.

## Kontakt

### Lizenzinformationen von Lizenzen mit Softwareschutz

#### Anzeige der Lizenzinformationen der internetbasierten Lizenzen

Folgende Informationen werden in der Lizenz-Identitätsdatei (\*.idn) gepeichert:



- Basis-Information

- ◆ Dateiname

Pfad- und Dateiname der Lizenz-Identitätsdatei.

- ◆ Lizenznummer

Hier wird die Lizenznummer angezeigt.

- ◆ Produktions-Datum

Hier wird angezeigt, wann die Lizenz produziert wurde.

- Server-Information

## Kontakt

Hier werden die Namen der Lizenzserver von vectorsoft aufgeführt. Die Reihenfolge entspricht der Reihenfolge, in der versucht wird die Server zu erreichen. Kommt zu einem Server keine Verbindung zustande, wird der nächste Server kontaktiert. Kann keiner der Server erreicht werden, ist die Lizenzabfrage gescheitert (siehe [Lizenz mit Softwareschutz \(Internet basierte Lizenzen\)](#)).

- Zeit-Information

- ◆ Letzte Anfrage

Zeitpunkt der letzten Abfrage an einen Lizenzserver. Dies ist der Zeitstempel des lokalen Rechners.

- ◆ Letzte Antwort

Zeitpunkt der letzten Antwort von einem Lizenzserver. Dies ist der Zeitstempel des Lizenzservers. Die Zeit wird an die lokale Zeitzone angepasst.

- Anfrage-Information

- ◆ Host-Name (Hash)

Der Hash-Wert des Rechnernamens. Zur Anonymisierung des Benutzers wird lediglich der Hash-Wert gespeichert. Durch ihn können keine Rückschlüsse auf den Rechnernamen erfolgen.

- ◆ Host Betriebssystem

Das Betriebssystem des Rechners.

- ◆ Host Arbeitsspeicher (MB)

Der in dem Rechner eingebaute Arbeitsspeicher.

- ◆ Host Prozessoren

Anzahl der logischen Prozessoren des Rechners.

- Antwort-Information

- ◆ Nächste Anfrage

Zeitpunkt der nächsten Lizenzabfrage.

- ◆ Intervall Verbindungs-Fehler

Abstand zur nächsten Lizenzabfrage, wenn keine Verbindung zu den Lizenzservern aufgebaut werden konnte.

- ◆ Intervall Anfrage-Fehler

Abstand zur nächsten Lizenzabfrage, wenn keine oder eine falsche Antwort vom Lizenzserver zurück gekommen ist.

- ◆ Gültigkeit

Die Lizenz-Identitätsdatei ist bis zu diesem Datum gültig. Der CONZEPT 16-Server kann bis zu diesem Datum betrieben werden, darüber hinaus können sich keine Benutzer mehr an einer Datenbank anmelden (siehe

## Kontakt

Lizenz mit Softwareschutz (Internetbasierte Lizenzen)).

♦ 1. Warnung

Die Anzahl der Tage vor Ablauf der Gültigkeit der Lizenz-Identitätsdatei zu dem die erste Warnung in die Protokolldatei geschrieben wird. Ist die Automatische E-Mail-Benachrichtigung beim CONZEPT 16-Server eingetragen, wird eine entsprechende E-Mail versendet.

♦ Anfrage-Fall

In diesem Eintrag wird das Ergebnis der letzten Lizenzabfrage angezeigt. Es werden folgende Resultate unterschieden (siehe auch Lizenz mit Softwareschutz):

- 0       Lizenzabfrage war ungültig
- 1       Erste Inbetriebnahme
- 2       Änderung der Konfiguration
- 3       Lizenzabfrage mit aktuellem Schlüssel
- 4 und 5 Lizenzabfrage mit altem Schlüssel

♦ Balance Lizenz Bewertung

der Lizenz.

♦ Balance Host Bewertung des

Systems.

Die Informationen im Bereich "Lizenz-Information" sind im Abschnitt Lizenzinformationen aus der Lizenzdatei beschrieben.

## Kontakt

**CONZEPT 16-Server - Installation des Servers unter Windows** Beschreibung der Installation des CONZEPT 16-Servers unter Windows

Der **CONZEPT 16-Server** wird über die CONZEPT 16 Installationsroutine eingerichtet. Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \Server. Über die Installationsroutine können die folgenden Komponenten für den Server unter Windows installiert werden:

- **Server (Windows)**

Diese Option beinhaltet den Datenbank-Server, das Lizenzinformationsprogramm und das Script-Utility.

- **Control-Center**

Mit dieser Option wird das Control-Center zur Administration des Datenbank-Servers installiert. Eine separate Installation des Control-Centers ist dann sinnvoll, wenn der Datenbank-Server vom aktuellen System aus nur administriert werden soll. Gerade wenn der Server auf einem Linux-System eingerichtet wurde, bietet diese Variante eine sehr komfortable Möglichkeit zur Server-Verwaltung. Das Control-Center beinhaltet ebenfalls den Log-Viewer.

### Installationsverlauf

Während der Installation des Datenbank-Servers über die Installationsroutine werden zusätzliche Dialoge zur Einrichtung des Servers angezeigt. Über das erste Konfigurationsfenster kann entschieden werden, ob der CONZEPT

16-Datenbank-Server als Dienst eingerichtet werden soll und somit beim Systemstart automatisch mit initialisiert wird (siehe auch Starten und Stoppen unter Windows). Ein Anmelden eines Benutzers an das Betriebssystem ist dann für den Serverbetrieb nicht notwendig. Wird der Server als Dienst eingerichtet, kann der Administrator über die zweite Option festlegen, dass der Server-Dienst nach der Installation automatisch gestartet werden soll.

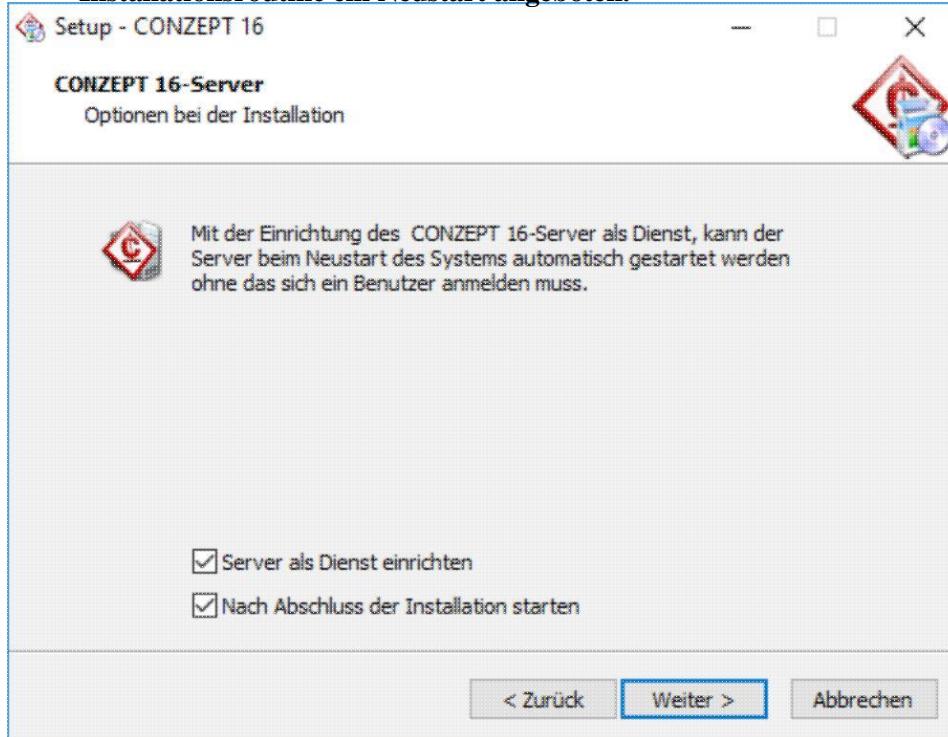
Falls der Server nicht als Dienst eingerichtet wird, ist die zweite Option nur aktiv, wenn das Control-Center ebenfalls zur Installation ausgewählt wurde. Der Datenbank-Server wird in diesem Fall nach der Installation durch das Control-Center im Detached-Mode gestartet. Dies ist dann auch bei jedem nächsten Start des Control-Centers der Fall. Diese Einstellung kann später im Control-Center noch angepasst werden.



Windows Server-Betriebssysteme räumen dem Dateicache für Dateifreigaben im Netzwerk in der Regel eine sehr hohe Priorität ein. Dies kann unter Umständen dazu führen, dass der Datenbank-Cache des CONZEPT 16-Servers benachteiligt wird und es zu Performance-Verlusten kommt. Um diese Priorisierung zu umgehen, kann in der Netzwerkkonfiguration des Betriebssystems die Option "Lastenausgleich durchführen" aktiviert werden. Ist diese Option zum Zeitpunkt der Server-Installation nicht aktiv, bietet die Installationsroutine dem Administrator über das zusätzliche Kontrollkästchen "Systemcache-Einstellung anpassen" die Möglichkeit, den Lastenausgleich automatisch aktivieren zu

## Kontakt

lassen. Nach der Installation wird bei Aktivierung der Option durch die Installationsroutine ein Neustart angeboten.



Für den Betrieb des CONZEPT 16-Servers stehen zwei Lizenzmodelle zur Verfügung.  
Die verwendete Lizenzierung wird in dem folgenden Dialog ausgewählt:

Es kann zwischen "Dongle-Lizenz installieren" (Lizenz mit Hardwareschutz (Dongle), es wird ein Dongle benötigt) und "Internetbasierte Lizenz installieren" (Lizenz mit Softwareschutz (Internetbasierte Lizenz), es wird eine Lizenzidentitätsdatei (c16\_?????.idn) benötigt) gewählt werden. Über die Schaltfläche [>>] können weitere Informationen zu den jeweiligen Lizenzmodellen angezeigt werden.

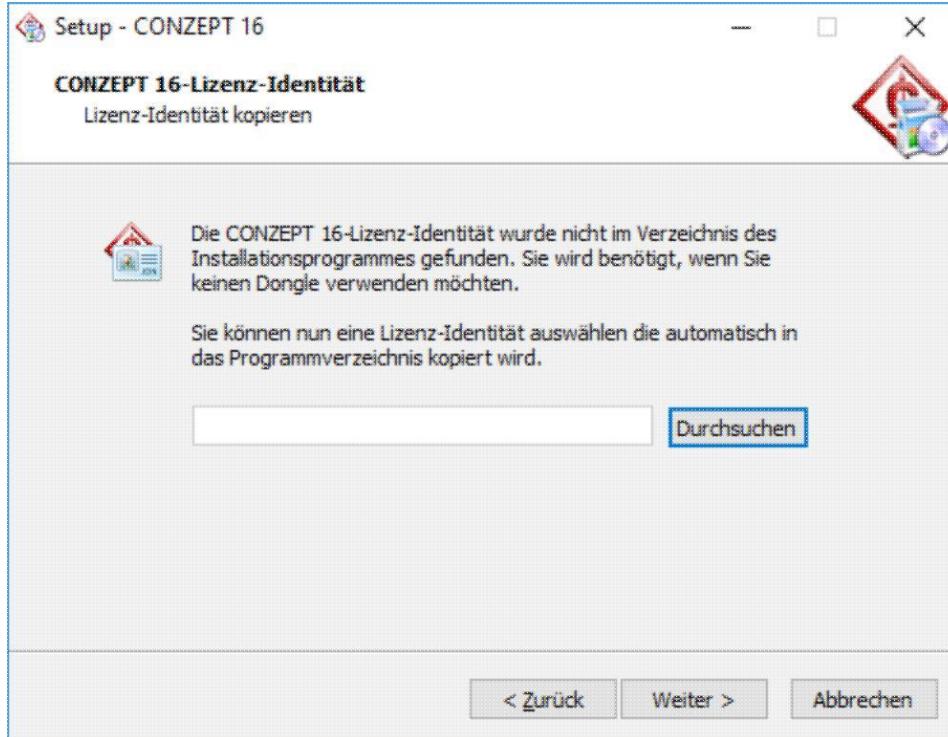
Anschließend wird der Speicherort der Lizenzdatei ausgewählt, welche die Lizenzausprägungen zu den Lizenznummern enthält. Die Datei wird dann im Verzeichnis für Konfigurationsdateien abgespeichert. Falls die Lizenzdatei noch nicht vorliegt, kann über den Link auch direkt die vectorsoft-Homepage angezeigt werden, um die aktuelle Lizenzdatei herunterzuladen.

**i** Dieses Fenster wird nicht angezeigt, falls eine Lizenzdatei bereits im Verzeichnis der Installationsroutine abgelegt wurde und somit automatisch übernommen werden kann. Falls eine Lizenzdatei aus dem Programmstand des Servers 5.1 übernommen wird, taucht dieses Fenster wiederum auf, damit eine Lizenzdatei der Version 5.2 angegeben werden kann, insofern diese nicht schon im alten Programmverzeichnis vorhanden war.

Wurde bei den Optionen der Lizenzinstallation "Internetbasierte Lizenz installieren" ausgewählt, kann in einem weiteren Dialog der Speicherort der Lizenz-Identitätsdatei angegeben werden. Die Datei wird dann ebenfalls im Verzeichnis für

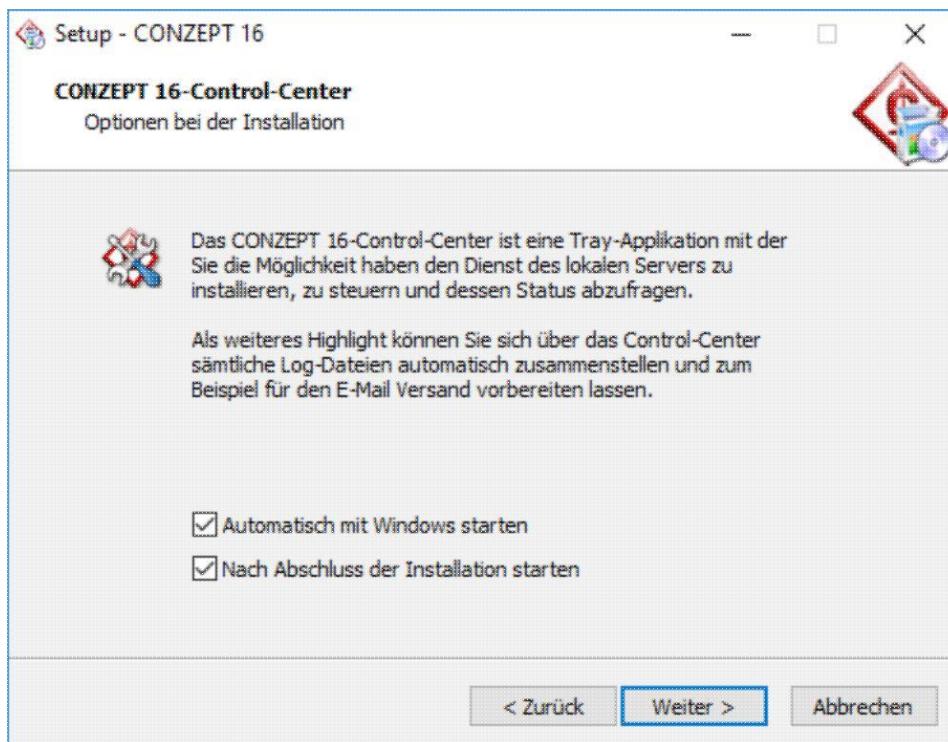
## Kontakt

**Konfigurationsdateien abgespeichert. Befindet sich diese Datei bereits im Verzeichnis der Installationsroutine, erfolgt keine Abfrage und die Datei wird in das korrekte Verzeichnis kopiert.**

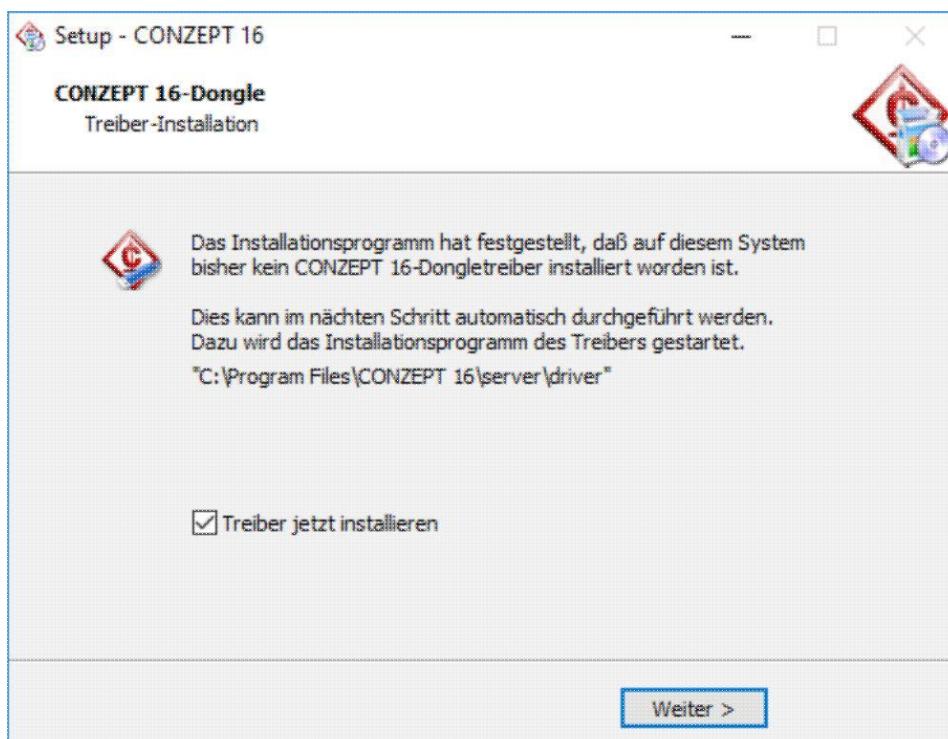


**Wurde bei der Installation auch das Control-Center ausgewählt, kann hier entschieden werden, ob das Control-Center so eingerichtet werden soll, dass es bei jeder Benutzeranmeldung mitgestartet wird. Diese Einstellung kann nach der Installation in den Einstellungen des Control-Centers noch angepasst werden. Mit der zweiten Option wird definiert, ob das Control-Center nach der Installation automatisch gestartet wird. Die zweite Option ist deaktiv, falls der Server im Anschluss durch das Control-Center im Detached-Mode gestartet werden soll.**

## Kontakt



Falls nach der Installation auf dem System kein Treiber für den CONZEPT 16 Lizenzdongle gefunden und keine internetbasierte Lizenz verwendet wird, kann die Installationsroutine die Treiberinstallation im Anschluss automatisch starten. Dazu wird die in diesem Fenster angebotene Option aktiviert. Die Installation des Dongles kann aber auch später mit der Datei c16\_dongle.exe aus dem Verzeichnis \driver gestartet werden. Erst nach der Installation des Dongle-Treibers darf der CONZEPT 16-Lizenzdongle auf das System aufgesteckt werden.



## Kontakt

### Installation des Dongle-Treibers

Zur Installation des Dongle-Treibers müssen zunächst die Lizenzvereinbarungen bestätigt werden.

In diesem Dialog muss die komplette Installation ausgewählt werden. Nur dann werden alle notwendigen Treiber für den Betrieb eines CONZEPT 16-Lizenzdongles installiert.

### Programmdateien

#### *Dateien des Servers*

c16_control.exe	<u>Control-Center</u>
c16_control_admin.exe	Administrator-Modul des Control-Centers
c16_icu_w32.dll	Unicode-Bibliothek für 32-Bit-Systeme
c16_icu_w64.dll	Unicode-Bibliothek für 64-Bit-Systeme
c16_lgbview.exe	<u>Log-Viewer</u>
c16_licinfo.exe	Programm zum Auslesen der Lizenzdatei
c16_serv_cmd_win.exe	<u>Script-Utility</u>
c16_serv_dba_w32.dll	Datei des Datenbank-Prozesses für 32-Bit-Systeme
c16_serv_dba_w64.dll	Datei des Datenbank-Prozesses für 64-Bit-Systeme
c16_serv_mgr_w32.dll	Datei des Manager-Prozesses für 32-Bit-Systeme
c16_serv_mgr_w64.dll	Datei des Manager-Prozesses für 64-Bit-Systeme
c16_serv_rpu_w32.dll	<u>Replikations Utility</u> für 32-Bit-Systeme
c16_serv_rpu_w64.dll	<u>Replikations Utility</u> für 64-Bit-Systeme
c16_serv_svc_win.dll	Datei des Service-Prozesses
c16_serv_vss_a32.dll	<u>Datei des VSS-Writers</u> für 32-Bit-Systeme (Windows Server 2003 / Windows Server 2008 / Windows Vista)
c16_serv_vss_a64.dll	<u>Datei des VSS-Writers</u> für 64-Bit-Systeme (Windows Server 2003 / Windows Server 2008 / Windows Vista)
c16_serv_vss_w32.dll	<u>Datei des VSS-Writers</u> für 32-Bit-Systeme ab Windows Server 2008 R2 / Windows 7
c16_serv_vss_w64.dll	<u>Datei des VSS-Writers</u> für 64-Bit-Systeme ab Windows Server 2008 R2 / Windows 7
c16_ssl_w32.dll	SSL-Bibliothek für 32-Bit-Systeme
c16_ssl_w64.dll	SSL-Bibliothek für 64-Bit-Systeme
c16.vra	Ressourcendatei
\driver	Treiber für den CONZEPT 16-Lizenz-Dongle
c16_dongle.exe	Installationsroutine für den Dongle-Treiber
\example	Beispieldateien

## Kontakt

area_backup.cmd	Skript für das CONZEPT 16-Script-Utility (Kopieren einer Datenbank)
backup_onoff.cmd	Skript für das CONZEPT 16-Script-Utility (Ein-/Ausschalten von <u>Sicherungsereignissen</u> )

-  Soll eine Internet-basierte Lizenz installiert werden, kann die notwendige Lizenz-Identitätsdatei in das Verzeichnis der Installationsroutine kopiert werden. Sie wird dann automatisch in das richtige Verzeichnis kopiert.
-  Die Lizenzinformationen werden aus der Lizenzdatei ausgelesen. Diese kann vor dem Start der Installation im Verzeichnis der Installationsroutine abgelegt werden. Von dort aus wird sie automatisch bei der Installation in das entsprechende Verzeichnis für Konfigurationsdateien kopiert.
-  Der Datenbank-Server entnimmt seine Konfiguration aus der Datenraumtabelle. Diese kann bei der Installation vordefiniert werden. Hierzu wird die Datei c16\_serv.ars vor dem Start der Installation in das Verzeichnis der Installationsroutine kopiert.

## Kontakt

### CONZEPT 16-Server - Starten und Stoppen unter Windows Starten und Stoppen des CONZEPT 16-Servers unter Windows

Der CONZEPT 16-Server kann als Dienst oder im Detached-Modus gestartet werden. Als Dienst startet der Datenbankserver im Systemkontext und ist somit unabhängig vom Anmeldestatus eines Benutzers. Im Detached-Modus wird der Datenbankserver im aktuellen Benutzerkontext gestartet. Der Datenbankserver ist dann abhängig vom aktuellen Benutzer und wird daher auch beim Abmelden des Benutzers wieder beendet.

-  **Der CONZEPT 16-Server wird generell ohne Benutzeroberfläche betrieben. Der Status des Servers und weitere Informationen können über das Control-Center und das Script-Utility ermittelt werden. Unabhängig davon, ob der Datenbankserver als Dienst oder im Detached-Mode betrieben werden soll, kann das Control-Center den Server über eine grafische Oberfläche Starten und Stoppen und auch als Dienst einrichten und entfernen.**
-  **Der CONZEPT 16-Server darf erst dann beendet werden, wenn keine Datenbank mehr geöffnet ist. Wird der Server trotzdem zuvor gestoppt, können offene Transaktionen nicht korrekt beendet und daher auch nicht mehr in die Datenbank übertragen werden.**

Wird der CONZEPT 16-Server als ein Server eines Hot-Standby-Paares gestartet, wird nur der Primärserver mit aufgestecktem Dongle gestartet. Der Sekundärserver muss ohne Dongle gestartet werden.

### Betreiben des Servers als Dienst

Der Datenbankserver kann unter Windows als Dienst eingerichtet werden. Dies geschieht entweder direkt bei der Serverinstallation oder nachträglich über das Control-Center. Das manuelle Starten des Servers entfällt dann und das Betriebssystem startet den Server unabhängig von einem Benutzerkontext automatisch beim Hochfahren des Systems.

Falls der Dienst gestoppt werden soll, kann dies entweder über das Control-Center oder die Diensteverwaltung des Betriebssystems gemacht werden. Das Entfernen des Dienstes aus dem Betriebssystem kann nur über das Control-Center durchgeführt werden. Bei der Deinstallation des CONZEPT 16-Servers wird der Dienst ebenfalls gestoppt und aus dem System entfernt.

-  **Wenn der CONZEPT 16-Server als Dienst eingerichtet ist, können auch die Betriebssystemkommandos "net start" und "net stop" zum Starten und Anhalten des Servers verwendet werden. Den Anweisungen muss lediglich der Name des Dienstes (`c16_server`) übergeben werden.**

### Starten des Servers im Detached-Mode

Der CONZEPT 16-Server kann im aktuellen Benutzerkontext gestartet werden. Der Start des Servers im Detached-Mode kann vom Control-Center durchgeführt werden. Ist beim Control-Center der Autostart-Modus und auch das automatische Initialisieren des CONZEPT 16-Servers eingestellt, wird der Datenbankserver immer bei einer Benutzeranmeldung mitgestartet.

## Kontakt

Wird das Control-Center über die Kommandozeile gestartet, kann der Datenbankserver aber auch automatisch im Detached-Mode mitgestartet werden. Dazu muss im Verzeichnis des CONZEPT 16-Servers das Control-Center mit der Anweisung `c16_control.exe -server_detached_start` aufgerufen werden. Ein automatisch ausgeführtes Skript mit diesem Befehl ist eine weitere Möglichkeit den CONZEPT 16-Server automatisch nach der Anmeldung eines Benutzers zu starten.



Innerhalb einer Terminalsession kann der CONZEPT 16-Server nicht im Detached-Modus gestartet werden.

Wird der Server-Prozess im Detached-Modus gestartet, besteht keine Verbindung zur Oberfläche des Betriebssystems. Dies hat zur Folge, dass, wird der Server bei geöffneter Datenbank heruntergefahren, der CONZEPT 16-Server nicht mehr die Gelegenheit hat, die Datenbank zu schließen. Wird die Datenbank erneut geöffnet, wird zur Sicherheit zunächst eine Diagnose mit Recover durchgeführt, die je nach Größe der Datenbank Zeit in Anspruch nimmt. Aus diesem Grund sollte der CONZEPT 16-Server immer als Dienst installiert werden. Der Detached-Modus sollte nur vorübergehend oder zu Testzwecken verwendet werden.

## Automatischer Start mit dem CONZEPT 16-Client

Wird der Server auf dem gleichen System betrieben wie der Client, kann er automatisch beim ersten Starten des Clients mitgestartet werden. Dazu muss in der [Konfigurationsdatei des Clients](#) nachfolgender Eintrag vorgenommen werden. Die Dateien des CONZEPT 16-Servers müssen sich dabei in dem angegebenen Verzeichnis befinden.

```
serverautostart = <Programmpfad des Servers>
```

Der CONZEPT 16-Server wird dann jedoch nicht in jedem Fall automatisch beim Client-Start mitgestartet. Falls auf dem System bereits ein Datenbankserver läuft, wird der Server nicht noch einmal gestartet. Wurde der Client innerhalb einer Terminalsession gestartet, so erfolgt auch kein automatischer Serverstart. Für den Fall dass der Server als Dienst eingerichtet ist, wird der Autostart auch generell unterbunden, unabhängig davon, ob der Dienst auch wirklich gestartet ist. Falls der angemeldete Benutzer keine ausreichende Berechtigung besitzt um den aktuellen Betriebszustand von Diensten zu überprüfen, wird der Server ebenfalls nicht gestartet.

## Kontakt

**CONZEPT 16-Server - Installation des Servers unter Linux** Beschreibung der Installation des CONZEPT 16-Servers unter Linux Siehe [Blog](#)

Die Programmdateien des **CONZEPT 16-Servers** unter Linux werden über die CONZEPT 16 Installationsroutine erstellt. Die Programmdateien und die Installationsroutine für den Lizenz-Dongle befinden sich anschließend im Installationsverzeichnis unter \Server\linux im komprimierten TAR-Archiv

c16\_server\_lnx.tar.gz. Dieses wird auf ein Linux-System transferiert und dort mit dem folgenden Kommando dekomprimiert und entpackt:

```
tar -zxvf c16_server_lnx.tar
```

Dies geht auch in einzelnen Schritten mit den Kommandos:

```
gzip -d c16_server_lnx.tar.gz tar -xvf c16_server_lnx.tar
```

### *Inhalt des Archivs*

c16_serv_svc_l32	Serviceprozess (Daemon) 32-Bit
c16_serv_svc_l64	Serviceprozess (Daemon) 64-Bit
c16_serv_mgr_l32	Managerprozess für 32-Bit
c16_serv_mgr_l64	Managerprozess für 64-Bit
c16_serv_dba_l32	Datenbankprozess für 32-Bit
c16_serv_dba_l64	Datenbankprozess für 64-Bit
c16_serv_cmd_l32	Script-Utility für 32-Bit
c16_serv_cmd_l64	Script-Utility für 64-Bit
c16_serv_rpu_l32	Replication Utility für 32-Bit
c16_serv_rpu_l64	Replication Utility für 64-Bit
c16.vra	Ressourcendatei
c16_serv	Init-Skript (SUSE)
c16_error	Liste der behobenen Fehler
\driver	Treiber für den CONZEPT 16-Lizenz-Dongle
sntl-sud-7.5.2.tar.gz	Dongle-Treiber für den Linux-Kernel 2.6
debian_7.5.1.tar.gz	Dongle-Treiber im Debian Binärpaketformat

### Installation des Dongle-Treibers

Wird eine internetbasierte Lizenz verwendet, muss der Dongle-Treiber nicht installiert werden.

Zu Beginn der Installation muss der Dongle-Treiber eingerichtet werden. Die Dongle-Treiber für den CONZEPT 16-Lizenz-Dongle werden ebenfalls als komprimierte TAR-Archive zur Verfügung gestellt und können wie oben beschrieben dekomprimiert und entpackt werden. Dabei ist zu beachten, dass der Treiber für den Kernel 2.6 mindestens SUSE Linux 10 SP 2 voraussetzt. Zur Installation sind folgende Schritte notwendig:

- Im Verzeichnis der entpackten Dateien das Installationsskript mit der Anweisung "sh sud\_install.sh" ausführen.

## Kontakt

- Lizenzvereinbarung bestätigen.
- Nach erfolgreicher Installation wird die Meldung "Sentinel USB Daemon installation was successful" ausgegeben. Anschließend kann der CONZEPT 16-USB-Dongle auf einen USB-Port aufgesteckt werden.

Bei korrekter Installation wird distributionsabhängig entweder bei jedem Neustart des Rechners der USB-Daemon automatisch mitgestartet oder er muss manuell geladen werden. Dies geschieht über die Anweisung "sh load\_daemon.sh start" aus dem Verzeichnis opt/safenet\_sentinel/common\_files/sentinel\_usb\_daemon.

Die Deinstallation des Dongle-Treibers kann über die Anweisung sh sud\_uninstall.sh vorgenommen werden. Dieses Skript befindet sich im mitgelieferten Archiv.

### Installation des CONZEPT 16-Servers

Zu den Dateien des CONZEPT 16-Servers wird die Lizenzdatei und eine eventuell schon vorhandene Datenraumtabelle kopiert. Anschließend kann der Server gestartet werden (siehe Starten und Stoppen des Servers unter Linux).



Der Server kann so eingerichtet werden, dass er beim Start des Betriebssystems automatisch mitgestartet wird. Details zum Boot-Konzept der verschiedenen Distributionen befinden sich in der entsprechenden Dokumentation der Distribution.

### Update des CONZEPT 16-Servers

Zum Update des CONZEPT 16-Servers wird dieser zunächst beendet. Dann können die neuen Programmdateien in das Verzeichnis des Servers entpackt und der Server in der neuen Version gestartet werden.

## Kontakt

**CONZEPT 16-Server - Starten und Stoppen unter Linux Starten und Stoppen des CONZEPT 16-Servers unter Linux Starten des Servers**

**Der Server wird auf allen 32- Bit Linux-Plattformen mit dem Programm c16\_serv\_svc\_l32 und auf allen 64-Bit Linux-Plattformen mit dem Programm c16\_serv\_svc\_l64 gestartet. Befindet sich der aktuelle Pfad nicht in der Path-Variable des Betriebssystems, muss ./ der Anweisung vorangestellt werden. Nach dem Starten des Service-Prozesses wird auf der Konsole folgendes ausgegeben:**

```
CONZEPT 16 Server Version 5.8.12 - Linux (C) Copyright by vectorsoft AG 1987-2020 All ri
```

**Der Service-Prozess wird unter Linux als Daemon gestartet, das heißt der Service-Prozess hängt sich direkt an den Init-Prozess des Betriebssystems an und ist somit von den Benutzer-Sessions unabhängig. Der Programmaufruf von c16\_serv\_svc\_l32 beziehungsweise c16\_serv\_svc\_l64 kehrt sofort zurück. Beim Starten kann aber zusätzlich die Option -wait angegeben werden. In diesem Fall kehrt der Aufruf erst zurück, wenn der Service-Prozess erfolgreich gestartet wurde.**

**Wird der CONZEPT 16-Server als ein Server eines Hot-Standby-Paars gestartet, wird nur der Primärserver mit aufgestecktem Dongle gestartet. Der Sekundärserver muss ohne Dongle gestartet werden.**

### Stoppen des Servers

**Der CONZEPT 16-Server wird mit der Anweisung "c16\_serv\_svc\_l32 -down" beziehungsweise "c16\_serv\_svc\_l64 -down" angehalten. Sind zu diesem Zeitpunkt noch Datenbanken geöffnet, erfolgt eine Sicherheitsabfrage. Die Abfrage wird nicht gestellt, wenn als Argument -force übergeben wird. Beim Stoppen des Servers kann ebenfalls das Argument -wait angegeben werden. Die Anweisung wartet dann maximal 20 Sekunden, bis der CONZEPT 16-Server vollständig heruntergefahren ist.**



**Die Argumente -wait und -force sollten in Skripts in jedem Fall verwendet werden, um eine unbeaufsichtigte Durchführung zu ermöglichen.**



**Der CONZEPT 16-Server darf erst dann beendet werden, wenn keine Datenbank mehr geöffnet ist. Wird der Server trotzdem gestoppt, können offene Transaktionen nicht beendet und somit auch nicht in der Datenbank gespeichert werden.**

## Kontakt

**Hinweise zum Betrieb auf virtuellen Maschinen**

**Besonderheiten beim Betrieb des Servers auf virtuellen Maschinen**

**Beim Betrieb des Datenbankservers in virtuellen Betriebssystemumgebungen sollten bei verschiedenen Aktionen bestimmte Regeln beachtet werden.**

**Anhalten und Wiederaufnehmen (Suspend & Resume)**

**Durch das Anhalten eines virtuellen Systems wird der Datenbankserver quasi "eingefroren". Dies gilt jedoch nicht für die TCP/IP-Verbindungen zu den Clients. Bereits durch ein Anhalten von wenigen Sekunden kann es bei einzelnen Clients zu einem Abbruch der Verbindung kommen. Sollte der Client zu diesem Zeitpunkt eine Transaktion geöffnet haben, wird diese abgebrochen. Je länger das System angehalten wird, umso wahrscheinlicher wird ein Kommunikationsabbruch der Clients. Aufgrund des TCP/IP-Keep-Alives sind nach spätestens zwei Stunden alle Client-Verbindungen beendet.**

**In den meisten Fällen können innerhalb der virtuellen Maschine Skripte definiert werden, die bei Suspend und Resume ausgeführt werden. In diesen Skripten kann dann der Datenbankserver entsprechend gestoppt und auch wieder gestartet werden. Dies ist auch für den Fall relevant, wenn die angehaltene Maschine nicht wiederaufgenommen werden kann, sondern neu gestartet werden muss (dadurch wird ein Datenbank-Rollback vermieden).**

**Das Anhalten im Hot-Standby-Betrieb ist extrem kritisch, da durch das Suspend des Datenbank-Masters unter Umständen das Standby-System aktiviert wird. Dabei führt dann das Wiederaufnehmen des Master-Prozesses zu einem Master-Master-Betrieb der Datenbank, bei dem die Clients gleichzeitig auf zwei verschiedenen Datenbankversionen arbeiten. Um dies zu verhindern, muss beim Anhalten des Masters der Datenbankserver in jedem Fall gestoppt werden.**

**Verschieben (Move)**

**Das Verschieben einer virtuellen Maschine entspricht dem Anhalten und Wiederaufnehmen über einen relativ kurzen Zeitraum (weniger als 1 Minute). Auch hierbei können Abbrüche der Clientverbindungen auftreten, ein Stopp des Datenbankservers ist beim Verschieben aber nicht zwingend notwendig.**

**Im Hot-Standby-Betrieb sollte bei einem Verschieben des Masters die automatische Aktivierung des Standby-Systems in jedem Fall ausgeschaltet werden, um eine versehentliche Aktivierung aufgrund eines Timeouts zu verhindern.**

## Kontakt

**CONZEPT 16-Server - Administration und Bedienung des Servers** Beschreibung der Administration und Bedienung des CONZEPT 16-Servers

Der CONZEPT 16-Server wird über seine Web-Oberfläche konfiguriert. Dazu zählen die Konfiguration des Servers und die Konfiguration der Datenbanken (siehe Konfiguration des Servers). Über den Standard-Client können Diagnose- und Optimierungsläufe durchgeführt werden. Die verschiedenen Möglichkeiten befinden sich im Hauptmenü des Standard-Clients unter Datenbank / Datenbankpflege.

Um den CONZEPT 16-Server zu starten und zu stoppen, beziehungsweise diesen als Dienst einzurichten und zu entfernen, wird das CONZEPT 16-Control-Center verwendet. Über das Control-Center können auch Statusinformationen und Log-Dateien des Servers abgeholt werden. Über das kommandozeilenbasierte Script-Utility können verschiedene Funktionen des Servers aufgerufen sowie Statusinformationen der Datenbanken abgefragt werden. Das Betrachten der Log-Dateien ist über den Log-Viewer sowie das Script-Utility möglich.



Beim Betrieb des Servers auf einer virtuellen Maschine sind einige Besonderheiten zu beachten. Hinweise dazu befinden sich im Abschnitt Hinweise zum Betrieb auf virtuellen Maschinen.

Log-Einträge sowie Diagnoseresultate werden ebenfalls am Ende dieses Kapitels beschrieben. Nachfolgend ein Überblick über die Abschnitte zur Administration und Konfiguration des Servers:

- Konfiguration des Servers
- Control-Center
- Script-Utility
- Log-Viewer
- Log-Einträge
- Diagnoseresultate

## Kontakt

### CONZEPT 16-Server - Konfiguration des Servers Beschreibung der Konfiguration des CONZEPT 16-Servers

Die Konfiguration des CONZEPT 16-Servers und die Verwaltung der Datenbanken erfolgt über eine Web-Oberfläche. Damit der Server über einen Web-Browser administriert werden kann, muss der dazu verwendete Kommunikations-Port in der Datenraumtabelle im Eintrag WebSvcPort angegeben werden. Nach der Installation des Servers wird der Port 4745 verwendet. Eine zusätzliche Software oder Plug-Ins für den Browser werden nicht benötigt.

Verfügt der Server über mehrere IP-Adressen, kann die Erreichbarkeit der Web-Administration auf bestimmte IP-Adressen beschränkt werden. Die entsprechenden IP-Adressen können durch Semikolon getrennt in den Eintrag WebSvcAddress der Datenraumtabelle eingetragen werden. Auf diese Weise kann eine Web-Administration aus bestimmten Netzen unterbunden werden.

Zur Administration des Servers mit einem Web-Browser muss nach dem Start des Browsers bei der Zieladresse der Name oder die IP-Adresse des Rechners angegeben werden, auf dem der CONZEPT 16-Server gestartet ist. Hinter der Zieladresse wird der Port angegeben:

**dbserver:4745**

Die Anzeige wird vom Browser automatisch auf <http://dbserver:4745> erweitert. Beim Verbinden mit dem Server wird nach einem Benutzer und einem Passwort gefragt. Der Benutzername lautet "admin" und das Standardpasswort "admin". Die Konfiguration kann ebenfalls aus dem Control-Center aufgerufen werden.



Das Standardpasswort muss geändert werden, um unbefugten Benutzern den Zugang zu verwehren.

Alle Eintragungen werden in der Datenraumtabelle (c16\_serv.ars, siehe auch Speicherorte von Konfigurationsdateien) des Servers gespeichert. Die Eintragungen in der Datenraumtabelle sind bei den entsprechenden Abschnitten mit angegeben. Diese Datei gliedert sich in mehrere Bereiche. Alle Eintragungen des Servers befinden sich im Bereich [AreaTable], die Einstellungen der Datenbanken befinden sich in einem [Area]-Bereich, wobei für jede Datenbank ein eigener Bereich definiert ist. Eine Beispiel-Konfigurationsdatei befindet sich im Abschnitt Beispiel - c16\_serv.ars.



Werden bei den Datenräumen Standardeinstellungen für bestimmte Einträge verwendet, werden diese Werte nicht in der Datenraumtabelle eingetragen. Da der Server die Datenraumtabelle nur beim Starten einliest, sind manuelle Änderungen an der Datei nur bei nicht aktivem Server sinnvoll. Änderungen über die Web-Administration können jederzeit vorgenommen werden. Die Änderungen wirken sich aber erst nach einem erneuten Öffnen der Datenbank aus.

Die Konfiguration gliedert sich in folgende Bereiche:

- Status
- Datenbank
  - ◆ Manager
  - ◆ Einstellungen
  - ◆ Backup

## Kontakt

- ◆ Hot-Standby
- ◆ Service
- ◆ Statistik
- Konfiguration

Die entsprechenden Bereiche können durch Anklicken der Namen aufgerufen werden.

Nach dem Aufruf der Web-Konfiguration wird die Status-Seite angezeigt.

## Kontakt

### CONZEPT 16-Server - Control-Center

#### Beschreibung der Funktionen des CONZEPT 16-Control-Centers

##### CONZEPT 16-spezifische

Siehe Dateien aktualisieren

##### (Blog)

Über das Control-Center können der Datenbankserver und alle anderen CONZEPT 16-Dienste verwaltet werden. Zudem ist es hier möglich sich Informationen zu den im Netz erreichbaren CONZEPT 16-Datenbankservern abzuholen. Auch Log-Dateien zu den Prozessen des Servers und zu den Datenbanken können aus dem Netzwerk über das Control-Center bequem und schnell abgeholt und angezeigt werden. Die Dokumentation des Control-Centers untergliedert sich in die folgenden Kapitel:

- Starten des Control-Centers - Einrichten der Control-Center Komponente
- Das Tray-Icon - Das Control-Center im Windows System-Tray
- Control-Center - Verwaltung des CONZEPT 16-Servers
  - ♦ Server - Abfragen von Server-Informationen
  - ♦ Dienste - Die Dienste-Überwachung des Control-Centers
  - ♦ Log-Dateien - Log-Dateien transferieren
  - ♦ Dateien - Verwaltung der Konfigurationsdateien
- Einstellungen - Konfiguration des Control-Centers
  - ♦ Allgemein - Autostart-Verhalten des Control-Centers
  - ♦ Server - Server-Definition
  - ♦ Log-Dateien - Definition der Log-Dateien

#### Starten des Control-Centers

Das Control-Center wird über das Programm c16\_control.exe gestartet. Diese ausführbare Datei befindet sich standardmäßig im Verzeichnis des CONZEPT 16-Servers. Zusätzlich zu diesem Modul befindet sich dort auch die Datei

c16\_control\_admin.exe. Diese führt alle Aufgaben des Control-Centers durch, welche administrative Systemrechte benötigen. Im Bedarfsfall wird dieses Modul dann automatisch gestartet. Wurde die Benutzerkontensteuerung (User Account Control - UAC) aktiviert, ist es notwendig vor dem Ausführen der ersten Administratorfunktion das Administratorpasswort einzugeben. Falls der Anwender bereits am System als Administrator angemeldet ist, muss nur eine entsprechende Sicherheitsmeldung bestätigt werden. Bis die Administratorfunktionen des Control-Centers freigeschaltet wurden, werden diese mit dem Sicherheitssymbol  gekennzeichnet.



Standardbenutzer können bei deaktivierter Benutzerkontensteuerung keine administrativen Funktionen ausführen. Dies wirkt sich ebenfalls auf die entsprechend mit einem Schild-Symbol gekennzeichneten Funktionen im Control-Center aus. Beim Ausführen einer dieser Funktionen, wie beispielsweise das Starten des Datenbank-Servers, erhält der Benutzer dann die Meldung, dass für den aktuellen Anwender keine Administratorenberechtigung existiert. Anschließend werden die entsprechenden Funktionen im Control-Center bis zum Neustart der Anwendung ausgeblendet.

Das Control-Center kann unter allen für den Server freigegebenen

## Kontakt

Windows-Betriebssystemen (vgl. [Systemvoraussetzungen](#)) gestartet werden. Dieses erscheint dann als Symbol im System-Tray. Wird beim Starten des Control-Centers auf der Kommandozeile die Option `-server_detached_start` angegeben, wird der CONZEPT 16-Server nach der Initialisierung des Control-Centers automatisch im Detached-Mode gestartet, sofern dieser nicht als Dienst auf dem System eingerichtet ist (siehe [Starten und Stoppen unter Windows](#)).

-  Da das Control-Center beispielsweise beim Start des Servers im Detached-Mode auf dessen Programmdateien zugreift, muss das Control-Center für diese Zwecke aus dem Server-Verzeichnis gestartet werden.

### Das Tray-Icon

Die Funktionen des Control-Centers werden über das Kontextmenü des Tray-Icons abgerufen.

Folgende Funktionen stehen dort zur Verfügung:



- Control-Center

Mit diesem Menüeintrag wird die Oberfläche des Control-Centers geöffnet. Das Control-Center kann auch durch einen Doppelklick auf das Tray-Icon geöffnet werden. Im Control-Center stehen die folgenden Funktionen zur Verfügung:

- Dienste

Über die Menüpunkte Datenbank Server, SOA-Service und Druck Prozessor können die entsprechenden Dienste gestartet, gestoppt oder neu gestartet werden, insofern diese auf dem lokalen System installiert sind. Falls der Datenbankserver nicht als Dienst eingerichtet ist, die Programmdateien aber trotzdem im Verzeichnis des Control-Centers liegen, kann der Server im Detached-Mode gestartet werden.

Wenn über das Control-Center CONZEPT 16-Dienste, beziehungsweise der Server im Detached-Mode gestartet werden, informiert ein kleines Popup-Fenster beim Tray-Symbol des Control-Centers über den aktuellen Status. Über fehlende Lizzenen und andere Probleme wird hier ebenfalls informiert.

- Einstellungen

## Kontakt

Über diesen Menüpunkt werden die Einstellungen des Control-Centers aufgerufen.

- Info

Mit diesem Menüpunkt wird ein Dialog mit dem Versionsstand des CONZEPT 16-Control-Centers ausgegeben. Unter dem Registerreiter "Parameter" im Informationsdialog wird der Speicherort der verwendeten Control-Center Konfigurationsdatei angezeigt.

- Beenden

Über diesen Menüpunkt wird das Control-Center beendet. Gestartete Dienste und der CONZEPT 16-Server laufen weiter. Der Menüpunkt ist ausgegraut, wenn der Konfigurationsdialog des Control-Centers geöffnet ist oder gerade eine Aktion (zum Beispiel das Starten des Servers) durchgeführt wird.

## Control-Center

Über das Control-Center können Serverinformationen, Dienste und Log-Dateien angezeigt werden. An der unteren Seite des Control-Centers können direkt die Einstellungen aufgerufen werden. Zudem ist es möglich die Anzeige zu aktualisieren. Mit [Schließen] wird das Control-Center geschlossen und ist dann wieder über das Tray-Icon erreichbar.

- Server

Auf der Registerseite "Server" des Control-Centers werden Server-Informationen angezeigt. Der entsprechende Server wird dazu aus der Liste ausgewählt. Die Server, die in der Liste angezeigt werden, können in den Einstellungen des Control-Centers definiert werden. Dabei ist es auch möglich Server anzugeben, die nicht auf dem lokalen System betrieben werden.

Alle verfügbaren Informationen zu dem angewählten Serversystem werden hier angezeigt. Falls Lizenz-Optionen im Einsatz sind, werden diese hier auch aufgeführt (zum Beispiel die Hot-Standby Option HSB).

Bei der Verwendung einer internetbasierten Lizenz werden ebenfalls die Informationen zum Ablauf der Lizenz und der nächsten Kontaktaufnahme zum Lizenzserver ausgegeben. Durch Drücken der Schaltfläche [Aktualisieren] wird der Kontakt zu einem Lizenz-Server aufgenommen. Auf diese Weise können die Einstellungen und die Verbindung zu den Lizenz-Servern überprüft werden.

Wurde bei der Eintragung des Servers ein Port angegeben, kann über die Schaltfläche [Administration] die Web-Administration des Servers aufgerufen werden. Ist kein Port angegeben, ist die Schaltfläche ausgegraut.

Über die Schaltfläche [Mail Test] kann die automatische E-Mail-Benachrichtigung des Servers getestet werden. Dabei wird gemäß der Server-Konfiguration eine Test-Mail versendet. Konnte die E-Mail versendet werden, kann das Protokoll des Managerprozesses direkt über den im eingeblendeten Statusfenster angezeigten Link aufgerufen werden. Falls der

## Kontakt

**Versand der E-Mail fehlschlägt, wird ein entsprechender Eintrag in der Log-Datei des Manager-Prozesses vorgenommen.**

**Die Schaltfläche [Lizenzserver] öffnet den Internet-Browser und fragt den Status aller vectorsoft-Lizenzserver ab. Das Ergebnis wird in den zurückgegebenen HTML-Seiten angezeigt.**

- Dienste

**Auf der Registerseite "Dienste" werden alle lokal installierten CONZEPT 16-Dienste in einer Liste angezeigt. Über das Symbol kann der Zustand des Dienstes festgestellt werden:**

Symbol Status	Durch Doppelklicken wird...
Dienst ist gestartet	... der Dienst angehalten.
Dienst ist angehalten	... der Dienst gestartet.
Dienst ist nicht installiert	... nicht gestartet. Der Dienst muss erst installiert werden.

**Die CONZEPT 16-Dienste können über die unter der Liste angezeigten Schaltflächen verwaltet werden. Folgende Funktionen stehen zur Verfügung:**

- “ **Installieren**

**Die Schaltfläche steht nur zur Verfügung, wenn der Dienst noch nicht installiert ist. Über die Schaltfläche kann der Dienst installiert werden. Die Installation erfolgt normalerweise über die Installationsroutine oder hier manuell. Falls der Server nicht als Dienst installiert ist, kann er über das Kontextmenü des Control Centers nur im Detached-Mode gestartet werden.**

- “ **Entfernen**

**Die Schaltfläche steht nur zur Verfügung, wenn der Dienst angehalten wurde. Mit dieser Schaltfläche wird der Dienst entfernt. Die Dateien bleiben in den entsprechenden Verzeichnissen erhalten. Der Dienst kann anschließend mit der Schaltfläche [Installieren] wieder installiert werden.**

- “ **Konfigurieren**

**Die Schaltfläche steht nur bei einem gestarteten SOA-Service zur Verfügung. Über die Schaltfläche wird die Liste der Task aufgerufen. In dieser Liste können Task hinzugefügt, geändert oder gelöscht werden.**

**Folgende Informationen können beim Erstellen eines neuen Task angegeben werden:**

- à Name - Name des Task (Eigenschaft SvcName)
- à Beschreibung - Beschreibung des Task (Eigenschaft SvcDescription)
- à Modus - Betriebsart (siehe Konfiguration des SOA-Service)

## Kontakt

- à Automatisch - Der Task startet automatisch mit dem Start des SOA-Service (siehe Konfiguration des SOA-Service)
- à Wiederherstellen - Der Task startet automatisch nach einem Absturz neu (siehe Konfiguration des SOA-Service)
- à 64-bit - Der Task wird in der 64-bit Version gestartet (siehe Konfiguration des SOA-Service)
- à Verzögerung - Der Task startet zeitverzögert mit dem Start des SOA-Service (siehe Konfiguration des SOA-Service)

Der Task wird angelegt, wenn der Dialog mit [Sichern] geschlossen wird.

**Der Name und der Modus eines bestehenden Task kann nicht geändert werden.**

### “ Starten

**Die Schaltfläche steht erst nach der Installation des Dienstes und wenn der Dienst noch nicht gestartet wurde, zur Verfügung. Über die Schaltfläche wird der Dienst gestartet.**

### “ Stoppen

**Die Schaltfläche steht nur bei einem gestarteten Dienst zur Verfügung. Durch Drücken der Schaltfläche wird der Dienst angehalten.**

### “ Neustart

**Die Schaltfläche steht nur bei einem gestarteten Dienst zur Verfügung. Durch Drücken der Schaltfläche wird der Dienst angehalten und erneut gestartet.**

## · Log-Dateien

Auf der Registerseite "Log-Dateien" können vom Control-Center Log-Dateien eines im Netzwerk verfügbaren CONZEPT 16-Servers zur Anzeige angefordert werden. Die anzufordernden Log-Dateien werden in den Einstellungen über verschiedene Profile definiert. Nach der Auswahl eines Profiles auf der Seite der Log-Dateien, werden durch Drücken der Schaltfläche [Datenbank-Info] Informationen zu der im Profil definierten Datenbank angezeigt. Ist keine Datenbank definiert, wird eine entsprechende Fehlermeldung ausgegeben. Über die Schaltfläche [Abholen] werden die Log-Dateien im Standardverzeichnis für Konfigurationsdateien unter Profiles\<Profilname> gespeichert. Die Namen der Protokolldateien werden um die Namen des Profils ergänzt. Das Verzeichnis wird anschließend geöffnet. Die Dateien können mit dem Log-Viewer betrachtet und ausgewertet werden. Mit der Schaltfläche [Anzeigen] werden die Log-Dateien übertragen und der Log-Viewer direkt gestartet. Hier kann dann über das Datei-Menü und die Öffnen-Schaltfläche des Log-Viewers zwischen den einzelnen Log-Dateien gewechselt werden.

Über die Schaltfläche [Versenden] werden die in dem Profil definierten Log-Dateien übertragen und der lokal installierte E-Mail Client aufgerufen. Die übertragenen Log-Dateien werden einer neu geöffneten E-Mail als Anhangdateien zugeordnet. Als Empfänger wird automatisch der vectorsoft-Support eingetragen. Der Empfänger sowie alle weiteren Daten der E-Mail können noch vor dem Versand vom Benutzer verändert werden.

## Kontakt

- Dateien

Über diese Registerseite können beliebige Konfigurationsdateien von CONZEPT 16 in die Verzeichnisse für Anwenderdaten kopiert werden. Dabei muss ein Benutzer mit den entsprechenden Schreibrechten nur den Dateiauswahl dialog über [Kopieren...] aufrufen und die gewünschten Dateien auswählen. Nach der Auswahl erfolgt eine Abfrage mit einer Information zum folgenden Kopiervorgang. Die Dateien werden dann entsprechend des Ablagesystems für Konfigurationsdateien kopiert.

Folgende Dateien können über das Control-Center kopiert werden:

c16\_xxxxxx.idn Lizenz-Identitätsdatei

c16.lic Lizenzdatei

c16\_serv.ars Datenraumtabelle

\*.cfg, \*.ini Konfigurationsdateien aller Clients (C16.CFG, C16\_APGL.CFG, c16\_debg.ini, c16\_ppcsvc.cfg, c16\_soa.cfg und die)

Konfigurationsdateien der Tasks)

c16.hst Hostnamen

Mit der Schaltfläche [Lizenzen] wird das Programm c16\_licinfo.exe gestartet.

Nähtere Informationen dazu befinden sich im Abschnitt Lizenzinformationen.

## Einstellungen

In den Einstellungen können neben den allgemeinen Optionen des Control-Centers Profile für Server und Log-Dateien definiert werden. Profile werden zum Beispiel benötigt, um dem Control-Center mitzuteilen, welche Server er im Hauptfenster des Control Centers anzeigen soll. Alle Konfigurationseinstellungen werden in der Datei c16\_control.ini im Arbeitsverzeichnis des Control Centers gesichert. Werden Änderungen an den Einstellungen vorgenommen wird zusätzlich die ursprüngliche Konfigurationsdatei als c16\_control.ini.bak gesichert. Das Sichern der Konfiguration kann über die Schaltfläche [Sichern] ausgelöst werden. Die Einstellungen teilen sich in die folgenden Bereiche auf.

- Allgemein

Über die Optionen im Bereich "Allgemein" kann festgelegt werden, ob das Control-Center automatisch beim Start von Windows (bei der Benutzeranmeldung) gestartet wird und ob das Control-Center den CONZEPT 16-Server im Detach-Mode ebenfalls automatisch mitstartet. Die zweite Option ist ausgegraut, wenn der Server auf dem System als Dienst installiert ist.

- Server

Im Bereich "Server" werden alle Serverprofile aufgelistet. Ein Serverprofil besteht aus einem Servernamen (nur für Darstellungszwecke) und einer IP-Adresse (zum Beispiel 127.0.0.1 für den lokalen Server) oder einem Rechnernamen. Die Serverprofile können im Hauptfenster des Control-Centers, ebenfalls unter "Server", in der Serverliste abgerufen werden, um die jeweiligen

## Kontakt

Serverinformationen anzuzeigen.

Neue Server werden mit [Hinzufügen] eingetragen. Der neue Name, beziehungsweise die IP-Adresse, darf dabei noch nicht in der Liste vorhanden sein. Der neue Eintrag darf die folgenden Zeichen nicht enthalten: ; =. Zulässig sind nur 7-Bit ASCII-Zeichen. Soll der Server über das Web administriert werden, muss im Eintrag "HTTP-Port" der entsprechende Port angegeben werden. Der Standardport ist 4745. Gegebenenfalls muss noch das Kennwort des CONZEPT 16-Servers in verschlüsselter oder unverschlüsselter Form angegeben werden. Das Kennwort wird in der c16\_control.ini gespeichert. Wird das verschlüsselte Kennwort eingetragen, steht es auch nicht im Klartext in der Datei. Das verschlüsselte Kennwort kann aus dem Eintrag "Password" der Datenraumtabelle kopiert werden. Vorhandene Einträge können über die Schaltfläche [Ändern] geändert und über [Entfernen] gelöscht werden.

- Log-Dateien

Im Bereich "Log-Dateien" werden Profile zum Abholen von unterschiedlichen Log-Dateien angelegt. Diese können dann im Hauptfenster des Control-Centers, ebenfalls unter "Log-Dateien", ausgewählt werden. Es können die binären Log-Dateien des Service-Prozesses, des Manager-Prozesses, der Datenbanken (\*.lgb), die Benutzerlog-Dateien (\*.lgu), die Datei des SOA-Service und dessen Tasks abgeholt werden. Standardmäßig ist ein Profil "Lokal" vorhanden. Bei diesem Profil werden die Log-Dateien des Service-Prozesses und des Manager-Prozesses berücksichtigt. Es können weitere Profile angelegt werden, in denen dann auch die Protokolldateien einer bestimmten Datenbank usw. enthalten sind.

Wird ein neues Profil mit [Hinzufügen] angelegt, muss ein Name angegeben werden. Der Name dient nur zur Anzeige und darf noch nicht als Profilname verwendet worden sein. Der Name darf die folgenden Zeichen nicht enthalten: ; = / \ : \* ? . Zulässig sind nur 7-Bit ASCII-Zeichen. In der Beschreibung kann ein beliebiger Text ohne die folgenden Zeichen angegeben werden: ; =. Im Bereich "Log-Dateien" muss ein Server ausgewählt werden. Anschließend können die unterschiedlichen Logs angekreuzt werden, die übertragen werden sollen. Soll das Log oder das Benutzerlog einer Datenbank übertragen werden, muss der symbolische Name der Datenbank aus der Datenbankliste ausgewählt, beziehungsweise in das Eingabefeld eingetragen werden. Hierbei werden in der Datenbankliste nur Datenbanken angezeigt, die mit einem symbolischen Namen beim entsprechenden Datenbankserver eingetragen wurden. Der symbolische Name der Datenbank darf die folgenden Zeichen nicht enthalten: ; =. Zulässig sind nur 7-Bit ASCII-Zeichen. Bestehende Profile können über die Schaltfläche [Ändern] verändert oder über [Entfernen] gelöscht werden.

Es können auch die Log-Dateien des SOA-Service abgeholt werden. Wird der Dienst angehakt wird die Datei c16\_soa.lgb abgeholt. Zusätzlich kann noch die Protokolldatei eines Tasks angegeben werden. Dazu muss der entsprechende Haken gesetzt und der Name des Task in der ComboBox ausgewählt werden.

## Kontakt

### CONZEPT 16-Server - Script-Utility

Beschreibung der Funktionen des CONZEPT 16-Script-Utilitys Siehe

#### [Blog](#)

Das Script-Utility ist ein Kommandozeilenprogramm, welches den Status von Datenbanken abfragen und auch setzen kann. Daneben kann es auch zur Anzeige der binären Log-Dateien verwendet werden. Das Script-Utility ist dabei für folgende Betriebssysteme verfügbar:

**c16\_serv\_cmd\_win.exe Windows 32- und 64-Bit**

**c16\_serv\_cmd\_l32              Linux 32-Bit**

**c16\_serv\_cmd\_l64              Linux 64-Bit**

Dem Programm müssen zur Durchführung der Kommandos bestimmte Argumente an der Kommandozeile übergeben werden. Die Kommandos sind für alle Betriebssysteme identisch. Die Argumente hängen vom jeweiligen Kommando ab.

**c16\_serv\_cmd\_\* <Kommando> [<Argumente>]**



Bei der Angabe von Argumenten wird zwischen Groß- und Kleinschreibung unterschieden.

Der Rückgabewert gibt Aufschluss darüber, ob und mit welchem Ergebnis das Kommando durchgeführt wurde. Wird das Script-Utility in einer Script-Datei verwendet, kann dieser Rückgabewert über den Errorlevel des Betriebssystems ermittelt und ausgewertet werden. Zusätzlich zum Rückgabewert wird auf der Kommandozeile auch ein Fehlerwert und eine entsprechende Information ausgegeben. Die ausgegebenen Fehlermeldungen entsprechen den [Log-Einträgen](#).

Durch die Angabe des Parameters **-quiet** oder **-quiet=inf** kann die Textausgabe von Informationen in der Kommandozeile unterdrückt werden. Durch die Angabe von **-quiet=err** werden zusätzlich Fehlerausgaben unterdrückt. Ebenso können Ausgabe forciert werden. Durch die Angabe des Parameters **-noquiet=err** werden Fehlermeldungen ausgegeben. Bei der Angabe von **-noquiet=inf** oder **-noquiet** werden ebenfalls Informationen ausgegeben.

Falls Kommandos ausgeführt werden, die den Datenbankserver betreffen, muss das Server-Kennwort angegeben werden. Wurde kein Server-Kennwort definiert, kann die Angabe entfallen oder das Argument bleibt leer. Mit '**-passwd=<Password>**' wird das Kennwort übertragen.

Folgende Kommandos können an das Script-Utility übergeben werden:

- **help**

Hilfe zum Script-Utility oder einem bestimmten Kommando anzeigen

- **version**

Versionsinformationen des Script-Utility anzeigen

- **status**

Status-Informationen des Servers anzeigen

## Kontakt

- **license**  
Lizenzinformationen zum Server anzeigen
- **receive**  
Log-Dateien beim Datenbank-Server abholen
- **decode**  
Anzeige des Inhalts von binären Log-Dateien
- **backup\_on**  
Datenbank in den Backup-Modus schalten
- **backup\_off**  
Backup-Modus einer Datenbank ausschalten
- **backup\_info**  
Abfrage des Backup-Modus
- **close**  
Schließen der Datenbank vor Ablauf der Schließverzögerung
- **down**  
Abmelden der Benutzer und schließen der Datenbank
- **lock\_on** Benutzerlogin-Sperre  
einschalten
- **lock\_off** Benutzerlogin-Sperre  
ausschalten
- **lock\_info** Datenbank-Sperre  
ermitteln
- **open\_info**  
Ermitteln, ob Datenbank offen ist oder nicht
- **clearhsblicense**  
Hot-Standby-Lizenzeintrag aus der Datenbank entfernen
- **area\_info**  
Ermittelt den Pfad zu einem Aliasnamen
- **area\_copy**  
Kopiert eine Datenbank in ein angegebenes Verzeichnis
- **diag**  
Datenbankdiagnose starten
- **perflog**  
Performance-Daten einer Datenbank aufzeichnen

## Kontakt

- **dump**

Speicherabbild der Systemdaten des Datenbankprozesses erzeugen

### Beispiel zur Verwendung des Script-Utility

Mit dem folgenden Skript kann eine CONZEPT 16-Datenbank im laufenden Betrieb kopiert werden. Die Datenbank wird dazu in den Backup-Modus versetzt und dann kopiert. Nach dem Kopiervorgang wird der Backup-Modus wieder zurückgesetzt.

```
@echo off if "%1" == "" goto Usage if "%2" == "" goto Usage c16_serv_cmd_win.exe area_copy %1 %2 -ti
```

## Kontakt

### CONZEPT 16-Server - Script-Utility

**area\_copy**

Siehe [Script-Utility](#)

**area\_copy** - Kopiert eine Datenbank in ein angegebenes Verzeichnis

**Syntax:**

```
c16_serv_cmd_win.exe area_copy <area alias> <destination path> -time=<hh:mm> [-ignore_active] [-w
```

**Funktion:**

Versetzt eine Datenbank in den **Backup-Modus**, kopiert sie in das angegebene Verzeichnis und schaltet den Backup-Modus wieder aus. Wird die Funktion auf eine geschlossene Datenbank durchgeführt, wird diese vom Datenbank-Server direkt im Backup-Modus geöffnet, falls sich ein Benutzer anmeldet. Die Gesamtdauer des Backup-Ereignisses reduziert sich dann um die seit dem Starten des Backup-Events verstrichene Zeit.

**Argumente:**

<b>&lt;area alias&gt;</b>	Symbolischer Name der Datenbank, die in den Backup-Modus versetzt werden soll.
<b>&lt;destination path&gt;</b>	Verzeichnis, in das die Datenbank kopiert werden soll.
<b>-time=&lt;hh:mm&gt;</b>	Dauer des Backup-Modus in Stunden und Minuten. Der Maximalwert beträgt 12 Stunden. Die Zeit sollte so gewählt werden, dass die Datenbank komplett kopiert werden kann.
<b>[-ignore_active]</b>	Ist dieser Schalter angegeben, kann auch eine Datenbank, die als Slave geöffnet ist kopiert werden.
<b>[-wait=&lt;int&gt;]</b>	Zeitspanne in Sekunden, die maximal gewartet wird, bis der Backup-Modus ein bzw. ausgeschaltet ist. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und die Anweisung abgebrochen. Wird keine Wartezeit angegeben, wird maximal 30 Sekunden gewartet. Als Wartezeit können bis zu 300 Sekunden (5 Minuten) angegeben werden.
<b>[-passwd=&lt;string&gt;]</b>	Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe <a href="#"><u>Konfiguration des Servers</u></a> ).
<b>[-options=&lt;y t&gt;]</b>	Über diesen Schalten können folgende Optionen beim Kopieren übergeben werden: <b>y</b> Ist die Datei bereits im Zielverzeichnis vorhanden, wird sie überschrieben. <b>t</b> Der Dateiname im Zielverzeichnis wird aus dem Datenbank-Dateinamen und dem aktuellen Datum (<name>_<yyyy>-<mm>-<dd>) gebildet. Es können beide Optionen gleichzeitig angegeben werden.

## Kontakt

### CONZEPT 16-Server - Script-Utility

#### area\_info

Siehe [Script-Utility](#)

**area\_info** - Ermittelt den Pfad zu einem Aliasnamen

Syntax:

```
c16_serv_cmd_win.exe area_info <area alias> [-ignore_active]
```

Ermittelt den vollständigen Pfad (ohne Dateierweiterung) und gibt ihn auf die Konsole aus.

Argumente:

<area alias>      Symbolischer Name der Datenbank.

**[-ignore\_activate]** Soll der Pfad der Datenbank ermittelt werden, die zur Zeit als Slave-Datenbank (siehe [Hot-Standby-Option](#)) geöffnet ist, muss dieser Schalter mit angegeben werden.

Der Datenbank-Pfad kann anschließend in einer Batch-Verarbeitung dazu verwendet werden, um die Datenbank zu packen o.ä. Bevor dies jedoch geschehen kann, muss der Backup-Modus (siehe [backup\\_on](#)) eingeschaltet werden.

Rückgabe:

- 0 - Die Datenbank ist als Slave geöffnet und der Schalter **-ignore\_active** ist nicht angegeben.
- 1 - Der Pfad wurde auf die Konsole ausgegeben.
- 2 - Fehler aufgetreten. Die Fehlerbeschreibung wurde auf die Konsole ausgegeben.



Da dieses Kommando speziell für den Einsatz in einer Batch-Verarbeitung vorgesehen ist, kann durch die Angabe der Option **-quiet=inf** erreicht werden, dass nur der Datenbank-Pfad und sonst nichts auf die Konsole geschrieben wird.

Beispiele:

Die Datenbank mit dem Alias "codelibrary" ist auf dem Primärsystem unter "c:\database\examples\codelibrary-1.ca1" und auf dem Sekundärsystem unter "c:\db\codelibrary-2.ca1" gespeichert. Die Datenbank auf dem Primärserver ist im Master-Betrieb, die auf dem Sekundärserver im Slave-Betrieb.

```
c16_serv_cmd_win.exe area_info codelibrary -quiet=inf
```

Auf dem Primär-System wird auf der Konsole c:\database\examples\codelibrary-1 ausgegeben. Wird die Anweisung auf dem Sekundärsystem ausgeführt, wird nichts ausgegeben.

```
c16_serv_cmd_win.exe area_info codelibrary -quiet=inf -ignore_active
```

Auf dem Primärsystem wird c:\database\examples\codelibrary-1 ausgegeben. Wird die Anweisung auf dem Sekundärsystem ausgeführt, wird c:\db\codelibrary-2 ausgegeben.

## Kontakt

Nach einem Rollenwechsel von Primär- und Sekundär-System (Primär-System=Slave-Datenbank, Sekundär-System=Master-Datenbank) führen die gleichen Befehle zu anderen Ausgaben:

```
c16_serv_cmd_win.exe area_info codelibrary -quiet=inf
```

Auf dem Primär-System erfolgt keine Ausgabe. Wird die Anweisung auf dem Sekundärsystem ausgeführt, wird c:\db\codelibrary-2 ausgegeben.

```
c16_serv_cmd_win.exe area_info codelibrary -quiet=inf -ignore_active
```

Auf dem Primärsystem wird c:\database\examples\codelibrary-1 ausgegeben. Wird die Anweisung auf dem Sekundärsystem ausgeführt, wird c:\db\codelibrary-2 ausgegeben.

## Kontakt

### CONZEPT 16-Server - Script-Utility

**backup\_info**

Script-Utility,

Siehe **backup\_on**,

backup\_off

**backup\_info** - Abfrage des Backup-Modus

Syntax:

```
c16_serv_cmd_win.exe backup_info <area alias> [-server=<Adresse>] [-passwd=<string>]
```

Ermittelt den Backup-Modus für die Datenbank.

Argumente:

**<area alias>** Symbolischer Name der Datenbank, für die die Information ermittelt werden soll.

**[-server=<Adresse>]** IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

**[-passwd=<string>]** Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe Konfiguration des Servers).

Rückgabe:

- 0 - Datenbank ist nicht im Backup-Modus.
- 1 - Datenbank ist im Backup-Modus.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

**backup\_off**

Script-Utility,

Siehe [backup\\_on](#),

[backup\\_info](#)

**backup\_off** - Backup-Modus einer Datenbank ausschalten

Syntax:

```
c16_serv_cmd_win.exe backup_off <area alias> [-wait=<int>] [-server=<Adresse>] [-passwd=<string>]
```

Setzt den Backup-Modus der Datenbank zurück.

Argumente:

<area alias>	Symbolischer Name der Datenbank, für die der Backup-Modus zurückgesetzt werden soll.
[-wait=<int>]	Zeitspanne in Sekunden, die maximal gewartet wird, bis das Kommando zurückkehrt. Ist die Datenbank bereits vor Verstreichen der angegebenen Zeitspanne nicht mehr im Backup-Modus, wird der Wartezustand abgebrochen. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und der Backup-Modus ist noch nicht beendet. Wird keine Wartezeit angegeben, wird maximal 300 Sekunden (5 Minuten) gewartet.
[-server=<Adresse>]	IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.
[-passwd=<string>]	Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe <u><a href="#">Konfiguration des Servers</a></u> ).

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

**backup\_on**

Script-Utility,

Siehe **backup\_off**,

backup\_info

**backup\_on** - Datenbank in den Backup-Modus schalten

Syntax:

```
c16_serv_cmd_win.exe backup_on <area alias> [-time=<hh:mm> [-wait=<int>] [-server=<Adresse>] [-pas
```

Versetzt eine Datenbank in den **Backup-Modus**. Wird die Funktion auf eine geschlossene Datenbank durchgeführt, wird diese vom Datenbank-Server direkt im Backup-Modus geöffnet, falls sich ein Benutzer anmeldet. Die Gesamtdauer des Backup-Ereignisses reduziert sich dann um die seit dem Starten des Backup-Events verstrichene Zeit.

Argumente:

**<area alias>** Symbolischer Name der Datenbank, die in den Backup-Modus versetzt werden soll.

**-time=<hh:mm>** Dauer des Backup-Modus in Stunden und Minuten. Der Maximalwert beträgt 12 Stunden.

**[-server=<Adresse>]** IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

**[-wait=<int>]** Zeitspanne in Sekunden, die maximal gewartet wird, bis das Kommando zurückkehrt. Ist die Datenbank bereits vor Verstreichen der angegebenen Zeitspanne im Backup-Modus, wird der Wartezustand abgebrochen. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und der Backup-Modus nicht gestartet. Wird keine Wartezeit angegeben, wird maximal 30 Sekunden gewartet. Als Wartezeit können bis zu 300 Sekunden (5 Minuten) angegeben werden.

**[-passwd=<string>]** Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe Konfiguration des Servers).

Wird während eines laufenden Backups ein weiterer Backup gestartet dessen Zeit größer als die Restzeit des laufenden Backups ist, wird der Backup-Modus entsprechend verlängert. Wird ein Backup mit 0 Minuten gestartet, wird ein laufender Backup beendet.

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

clearhsblicense

Siehe [Script-Utility](#)

clearhsblicense - Hot-Standby-Lizenzeintrag aus der Datenbank entfernen

Syntax:

```
c16_serv_cmd_win.exe clearhsblicense <area name>
```

Funktion:

Entfernt den Lizenzeintrag der Hot-Standby-Datenbank aus dem Datenbank-Header. Dieser Lizenzeintrag ermöglicht 60 Tage lang die Nutzung der Slave-Datenbank als Master, ohne die Lizenz vom Primärserver. Wurde der Eintrag erfolgreich entfernt, wird die Meldung Database header changed successfully ausgegeben.

Argument:

<area name> Vollständiger Dateiname der Datenbank, in der der Hot-Standby-Lizenzeintrag entfernt werden soll.

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

Weiterhin können Fehlerkonstanten aus dem Bereich der [externen Dateien](#) zurückgegeben werden.

Beispiel:

Lizenzeintrag der Datenbank "d:\Database\MyDatabase.ca1" entfernen.

```
c16_serv_cmd_win.exe clearhsblicense d:\Database\MyDatabase.ca1
```

## Kontakt

### CONZEPT 16-Server - Script-Utility

close

Siehe [Script-Utility](#)

close - Schließen der Datenbank vor Ablauf der Schließverzögerung

Syntax:

```
c16_serv_cmd_win.exe close <area alias> [-wait=<int>] [-passwd=<string>] [-server=<Adresse>]
```

Der Datenbankprozess wird beendet, d. h. die Datenbank wird vor Ablauf der Schließverzögerung geschlossen.



Es dürfen keine Benutzer angemeldet sein.

Argumente:

<area alias>	Symbolischer Name der Datenbank, die geschlossen werden soll.
[-wait=<int>]	Zeitspanne in Sekunden, die maximal gewartet wird, bis das Kommando zurückkehrt. Ist die Datenbank bereits vor Verstreichen der angegebenen Zeitspanne geschlossen, wird nicht länger gewartet. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und die Datenbank ist noch nicht geschlossen. Wird keine Wartezeit angegeben, wird maximal 300 Sekunden (5 Minuten) gewartet.
[-passwd=<string>]	Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe <a href="#"><u>Konfiguration des Servers</u></a> ).
[-server=<Adresse>]	IP-Adresse oder Name des Servers auf dem der CONZEPT 16-Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

decode

Siehe [Script-Utility](#)

decode - Anzeige des Inhalts von binären Log-Dateien

Syntax:

```
c16_serv_cmd_win.exe decode <blog1> [<blog2>] [-start=<date>] [-end=<date>] [-days=<int>] [-recor
```

Anzeige von bis zu zwei binären Log-Dateien, beginnend vom jüngsten Eintrag in chronologischer, absteigender Reihenfolge, mit zeitlicher Eingrenzung und Filterung der Einträge. Siehe auch [Log-Viewer](#). Falls im Verzeichnis der Log-Datei neben der <Datenbankname>.lgb auch das Archivlog <Datenbankname>.lga (siehe [Architektur des Servers](#)) liegt, wird das gesamte Log aus Standard-Log und Archiv-Log betrachtet.

Argumente:

<blog1>	Vollständiger Dateiname der ersten binären Logdatei. Die Anzeige erfolgt chronologisch absteigend.
[<blog2>]	Vollständiger Dateiname einer weiteren binären Logdatei. <blog1> und <blog2> werden in eine chronologische Reihenfolge gebracht.
[-start=<date>]	Anfangswert für die Datumseingrenzung in der Form yyyy-mm-dd. Es werden nur Log-Einträge angezeigt, die nicht älter sind, wie das angegebene Datum.
[-end=<date>]	Endwert für die Datumseingrenzung in der Form yyyy-mm-dd. Es werden nur Log-Einträge angezeigt, die nicht neuer sind, wie das angegebene Datum.
[-days=<int>]	Ist dieses Argument angegeben, hat es Vorrang vor "start" und "end". Es werden keine Einträge angezeigt, die älter als <int> Tage sind, ausgehend vom neuesten Eintrag in der Log-Datei.
[-records=<int>]	Ist dieses Argument angegeben, hat es Vorrang vor "days", "start" und "end". Es werden die letzten <int> Einträge, ausgehend vom neuesten Eintrag im binären Log angezeigt. Der höchstmögliche Wert für diese Option beträgt 500.000.
[-filterex=<string>]	Standardmäßig werden keine Datensätze ausgefiltert. Durch Angabe dieses Argumentes können Einträge bestimmten Typs und Klasse verborgen werden. Für <string> können folgende Zeichen kombiniert werden: "uhgiwed". Dabei werden dann folgende Einträge nicht angezeigt: <ul style="list-style-type: none"><li><b>u</b> Einträge des Typs User</li><li><b>h</b> Einträge des Typs Hot-Standby</li><li><b>g</b> Einträge, die weder vom Typ User noch vom Typ Hot-Standby sind</li><li><b>i</b> Einträge der Klasse Information</li><li><b>w</b> Einträge der Klasse Warnung</li><li><b>e</b> Einträge der Klasse Fehler</li><li><b>d</b> Einträge der Klasse Debug</li></ul>

## Kontakt

**[-charset=<string>]** Mit diesem Parameter kann der Zeichensatz bestimmt werden, der vom Script-Utility bei der Dekodierung der Log-Einträge verwendet wird. Da Texteinträge von Benutzerlog-Dateien im Windows-Zeichensatz gespeichert werden, ist es unter Umständen für Konsolen-Ausgaben erforderlich, die Ausgabe in den OEM-Zeichensatz zu wandeln, damit Umlaute auch auf der Konsole korrekt dargestellt werden. Wird der Zeichensatz nicht definiert, wird der ANSI-Zeichensatz verwendet. Folgende Angaben sind bei diesem Schalter zulässig:

**ansi** Dekodieren der Logdatei im Windows-Zeichensatz (ISO Latin I)

**oem** Dekodieren der Logdatei im OEM-Zeichensatz (PC-Zeichensatz)

**Rückgabe:**

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

**Beispiele:**

Alle Einträge der Manager-Logdatei in einer ASCII-Datei speichern.

```
c16_serv_cmd_win.exe decode c16_serv_mgr.lgb > manager.txt
```

Alle Einträge anzeigen, die zwischen dem 1. Juli 2007 und dem 3. Juli 2007 geschrieben wurden.

```
c16_serv_cmd_win.exe decode c16_serv_mgr.lgb -start=2007-07-01 -end=2007-07-03
```

Wie zuvor, jedoch werden alle Einträge, die nur informativer Natur sind ausgeklammert.

```
c16_serv_cmd_win.exe decode c16_serv_mgr.lgb -start=2007-07-01 -end=2007-07-03 -filterex=i
```

## Kontakt

### CONZEPT 16-Server - Script-Utility

diag

Siehe [Script-Utility](#),

[Datenbankdiagnose](#)

diag - Diagnose der Datenbank starten

Syntax:

```
c16_serv_cmd_win.exe diag <area alias> [-server=<Adresse>] [-passwd=<string>]
```

Startet die **Diagnose** einer Datenbank. Wird die Funktion auf eine geschlossene Datenbank durchgeführt, wird diese vom Datenbank-Server geöffnet. Wie auch bei der **automatischen Diagnose** wird die Diagnose vom Benutzer "<SYSTEM>" durchgeführt.

Argumente:

<area alias>                   Symbolischer Name der Datenbank, die diagnostiziert werden soll.

[-server=<Adresse>] IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

[-passwd=<string>] Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe [Konfiguration des Servers](#)).

Das **Diagnoseresultat** wird in der Datei <Datenbankname>\*.dgn im Verzeichnis der Datenbank abgelegt.

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

down

Siehe [Script-Utility](#)

down - Abmelden der Benutzer und schließen der Datenbank

Syntax:

```
c16_serv_cmd_win.exe down <area alias> \-passwd=<string> [-wait=<int>] [-server=<Adresse>]
```

Alle Benutzer der Datenbank werden abgemeldet und die Datenbank wird geschlossen.

Argumente:

<area alias>	Symbolischer Name der Datenbank, die geschlossen werden soll.
-passwd=<string>	Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe Konfiguration des Servers).
[-wait=<int>]	Zeitspanne in Sekunden, die maximal gewartet wird, bis das Kommando zurückkehrt. Ist die Datenbank bereits vor Verstreichen der angegebenen Zeitspanne geschlossen, wird nicht länger gewartet. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und die Datenbank ist noch nicht geschlossen. Wird keine Wartezeit angegeben, wird maximal 300 Sekunden (5 Minuten) gewartet.

[-server=<Adresse>] IP-Adresse oder Name des Servers auf dem der CONZEPT 16-Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.



Auf dem Server muss ein Server-Kennwort vergeben sein. Dieses muss zwingend zur Ausführung angegeben werden.

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

**dump**

Siehe Script-Utility

**dump** - Speicherabbild des Datenbankprozesses erzeugen

**Syntax:**

```
c16_serv_cmd_win.exe dump -shmid=<string>
```

**Es wird ein Speicherabbild der Systemdaten des Datenbankprozesses erzeugt.**

**Anwendungsdaten sind im Speicherabbild nicht enthalten.**

**Argumente:**

**<shmid> Shared-Memory-ID / Prozess-ID. Diese Information kann dem Datenbanklog entnommen werden.**

**Das Speicherabbild wird im Pfad der Datenbank nach dem Muster**

**<Datenbankname>.<Datum>\_<Uhrzeit> erzeugt.**

**Rückgabe:**

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

**help**

Siehe [Script-Utility](#)

**help - Hilfe zum Script-Utility oder einem bestimmten Kommando anzeigen**

**Syntax:**

```
c16_serv_cmd_win.exe help [<Kommando>]
```

**Ohne Angabe des <Kommando> wird eine Übersicht aller vorhandenen Kommandos angezeigt. Mit Angabe des <Kommando> wird die Befehlszeile und eine Beschreibung des angegebenen Kommandos angezeigt.**

## Kontakt

### CONZEPT 16-Server - Script-Utility

license

Siehe Script-Utility

license - Lizenzinformationen zum Server anzeigen

Syntax:

```
c16_serv_cmd_win.exe license [-server=<Adresse>]
```

Es werden Informationen zur Lizenz des Servers angezeigt. Ist kein Server angegeben (IP-Adresse oder Name), beziehen sich die Informationen auf den lokalen Server.

- Release License Versionsnummer der Lizenz.
- License Lizenz-Klartext (z. B. "CE100000MU/H")
- User Limit Benutzerlimit
- Edition Edition (z. B. "Enterprise")
- Options Lizenz-Optionen (z. B. "HSB")

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

#### lock\_info

Script-Utility,

Siehe lock\_on,

lock\_off

**lock\_info** - Benutzerlogin-Sperre ermitteln

Syntax:

```
c16_serv_cmd_win.exe lock_info <area alias> [-server=<Adresse>] [-passwd=<string>]
```

Ermittelt, ob eine Datenbank gesperrt ist.

Argumente:

**<area alias>** Symbolischer Name der Datenbank.

**[-server=<Adresse>]** IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

**[-passwd=<string>]** Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe Konfiguration des Servers).

Rückgabe:

- 0 - Datenbank ist nicht gesperrt.
- 1 - Datenbank ist gesperrt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

#### lock\_off

Script-Utility,

Siehe lock\_on,

lock\_info

#### lock\_off - Benutzerlogin-Sperre ausschalten

Syntax:

```
c16_serv_cmd_win.exe lock_off <area alias> [-wait=<int>] [-server=<Adresse>] [-passwd=<string>]
```

Setzt die Login-Sperre für eine Datenbank zurück.

Argumente:

<area alias>	Symbolischer Name der Datenbank, die entsperrt werden soll.
[-wait=<int>]	Zeitspanne in Sekunden, die maximal gewartet wird, bis das Kommando zurückkehrt. Ist die Datenbank bereits vor Verstreichen der angegebenen Zeitspanne entsperrt, wird der Wartezustand abgebrochen. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und die Datenbanksperre ist noch nicht entfernt. Wird keine Wartezeit angegeben, wird maximal 300 Sekunden (5 Minuten) gewartet.
[-server=<Adresse>]	IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.
[-passwd=<string>]	Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe <u>Konfiguration des Servers</u> ).

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

#### lock\_on

[Script-Utility](#),

Siehe [lock off](#),

[lock info](#)

#### lock\_on - Benutzerlogin-Sperre einschalten

Syntax:

```
c16_serv_cmd_win.exe lock_on <area alias> [-wait=<int>] [-server=<Adresse>] [-passwd=<string>]
```

Setzen der Login-Sperre für eine Datenbank, um das neue Einloggen von Benutzern zu verhindern. Die Login-Sperre bleibt bestehen, bis sie mit der Anweisung **lock\_off** aufgehoben oder der CONZEPT 16-Server neu gestartet wird.

Argumente:

**<area alias>** Symbolischer Name der Datenbank, die gesperrt werden soll.

**[-wait=<int>]** Zeitspanne in Sekunden, die maximal gewartet wird, bis das Kommando zurückkehrt. Ist die Datenbank bereits vor Verstreichen der angegebenen Zeitspanne gesperrt, wird der Wartezustand abgebrochen. Reicht die Wartezeit nicht aus, wird der Fehlerwert 2 zurückgegeben und die Datenbanksperre ist noch nicht eingerichtet. Wird keine Wartezeit angegeben, wird maximal 300 Sekunden (5 Minuten) gewartet.

**[-server=<Adresse>]** IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

**[-passwd=<string>]** Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe [Konfiguration des Servers](#)).

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

#### open\_info

Siehe [Script-Utility](#)

**open\_info** - Ermitteln, ob Datenbank offen ist oder nicht

**Syntax:**

```
c16_serv_cmd_win.exe open_info <area alias> [-server=<Adresse>] [-passwd=<string>] [-mode=slave]
```

Ermittelt, ob eine Datenbank zur Zeit geöffnet ist.

**Argumente:**

**<area alias>** Symbolischer Name der Datenbank.

**[-server=<Adresse>]** IP-Adresse oder Name des Servers auf dem der CONZEPT 16 Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

**[-passwd=<string>]** Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe [Konfiguration des Servers](#)).

**[-mode=slave]** Ist diese Option angegeben, wird geprüft, ob die Datenbank als Hot-Standby-Slave geöffnet ist.

**Rückgabe:**

- 0 - Datenbank ist geschlossen bzw. nicht als Slave-Datenbank geöffnet.
- 1 - Datenbank ist offen bzw. als Slave-Datenbank geöffnet.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

perflog

Siehe **Script Utility**,  
Blog

perflog - Performance-Daten einer Datenbank aufzeichnen

Syntax:

```
c16_serv_cmd_win.exe perflog <area alias> -mode=<on|off> [-server=<Adresse>] [\‐passwd=<string>]
```

Aufzeichnung der Performance-Log der Datenbank (de-)aktivieren.

Argumente:

<area alias> Symbolischer Name der Datenbank, deren Performance-Daten aufgezeichnet werden sollen.

[-mode=<on|off>] Performance-Log aktivieren (on) oder deaktivieren (off).

[-server=<Adresse>] IP-Adresse oder Name des Servers auf dem der CONZEPT

16-Server installiert ist. Wird die Angabe weggelassen, wird der lokale Server angesprochen.

[\‐passwd=<string>] Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe Konfiguration des Servers).



Die Aufzeichnung wird beim Schließen der Datenbank automatisch deaktiviert. Läuft die Datenbank im 24-Stunden-Betrieb, sollte das Performance-Log nach Verwendung wieder deaktiviert werden, da bei Erzeugung des Performance-Logs eine große Datenmenge anfallen kann.

Wird die Aufzeichnung aktiviert, erzeugt der Datenbankserver im Verzeichnis der Datenbank eine Datei Namens <Datenbankname>\_perf\_YYYYMMDD.csv. Bei einem Datumswechsel wird automatisch eine neue Datei generiert. In den Zeilen werden jeweils die Abweichungen zur vorherigen Zeile protokolliert. Die Datei besitzt folgende Spalten:

TIME	Uhrzeit für diese Spalte
USR	Anzahl der Benutzer (aktive Datenbankverbindungen)
REQ	Anzahl der Benutzeranfragen
TRC	Anzahl der Transaktionen
LCK	Anzahl der Sperrkonflikte
RBT	Anzahl der Verklemmungen (Deadlocks)
SEG_RC	Logische Segment-Leseoperationen im Cache (KB)
SEG_R	Physikalische Segment-Leseoperationen (KB)
SEG_WC	Logische Segment-Schreiboperationen im Cache (KB)
SEG_W	Physikalische Segment-Schreiboperationen (KB)
NET_IO	Netzwerk-I/O (KB)
HSB_IO	Hot-Standby-I/O (KB)
CA_R	Leseoperationen auf der Datenbank (KB)
CA_W	Schreiboperationen auf der Datenbank (KB)

## **Kontakt**

**TRS\_R** Leseoperationen in der .trs-Datei (KB)  
**TRS\_W** Schreiboperationen in der .trs-Datei (KB)  
**TL\_R** Leseoperationen in den Transaktionslogs (KB)  
**TL\_W** Schreiboperationen in den Transaktionslogs (KB)  
**DISK\_R** Summe der Leseoperationen (KB)  
**DISK\_W** Summe der Schreiboperationen (KB) **DISK\_IO**  
Summe der Schreib-/Leseoperationen (KB) Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

receive

Siehe [Script-Utility](#)

receive - Log-Dateien beim Datenbank-Server abholen

Syntax:

```
c16_serv_cmd_win.exe receive <path> [-area=<string>] [-passwd=<string>] [-server=<Adresse>] [-opt
```

Überträgt Log-Dateien vom Datenbank-Server an einen beliebigen Speicherort.

Argumente:

<path>	Zielpfad für die Log-Dateien.
[-area=<string>]	Aliasname der Datenbank.
[-passwd=<string>]	Passwort für den Server. Das Kennwort kann verschlüsselt oder unverschlüsselt angegeben werden (siehe <a href="#"><u>Konfiguration des Servers</u></a> ).
[-server=<Adresse>]	Adresse des Datenbank-Servers. Wird dieser Parameter nicht angegeben, wird der lokale Server verwendet.
[-type=<string>]	Log-Dateien, die angefordert werden sollen. Die einzelnen Typen können kombiniert werden. Wird kein Parameter angegeben, werden automatisch die Log-Dateien des Service- und des Manager-Prozesses übertragen. s Log-Datei des Service-Prozesses m Log-Datei des Manager-Prozesses d Log-Datei des Datenbank-Prozesses u Benutzerlog-Datei
[-options=<string>]	Zusätzliche Optionen y Bei Angabe dieser Option werden schon vorhandene Log-Dateien überschrieben. Dies ist ansonsten nicht der Fall. x Bei Verwendung dieser Option wird die gesamte Übertragung bei einem Übertragungsproblem abgebrochen. Ansonsten wird einfach mit der nächsten Log-Datei fortgefahrene

Rückgabe:

- 1 - Die Log-Datei(en) wurde(n) korrekt übertragen.
- 2 - Fehler aufgetreten.

## Kontakt

### CONZEPT 16-Server - Script-Utility

status

Siehe Script-Utility

status - Status-Informationen des Servers anzeigen

Syntax:

```
c16_serv_cmd_win.exe status [-server=<Adresse>]
```

Die Informationen des Servers werden angezeigt. Wird kein Server (IP-Adresse oder Name) angegeben, beziehen sich die Informationen auf den lokalen Server.

- Release SVC Versionsnummer des Service-Prozesses.
- Release MGR Versionsnummer des Manager-Prozesses.
- Status run (Server läuft), down (Server läuft nicht).
- Init Error Fehlerwert, falls beim Starten des Servers ein Fehler auftrat.
- Mode service (Server läuft als Dienst), detached (Server läuft detached).
- Areas closed (keine Datenbanken offen), open (Mindestens eine Datenbank ist offen).

Rückgabe:

- 0 - Server läuft nicht.
- 1 - Server läuft.
- 2 - Fehler aufgetreten.

## **Kontakt**

### **CONZEPT 16-Server - Script-Utility**

**version**

**Siehe Script-Utility**

**version - Versionsinformationen des Script-Utility anzeigen**

**Syntax:**

```
c16_serv_cmd_win.exe version
```

**Anzeige der Version des Script-Utility.**

## Kontakt

### CONZEPT 16-Server - Log-Viewer

#### Beschreibung der Funktion und Bedienung des CONZEPT 16-Log-Viewers

Siehe [Log-Dateien](#)  
[\(Blog\)](#)

Der CONZEPT 16-Server und der CONZEPT 16-SOA-Service legen Log-Dateien in einem binären und komprimierten Format ab. Um die Log-Dateien zur Ansicht zu öffnen wird daher der CONZEPT 16-Log-Viewer benötigt. Der Log-Viewer ermöglicht die Anzeige von bis zu zwei binären Log-Dateien in chronologischer Reihenfolge mit zeitlicher Eingrenzung und Filterung der Einträge (siehe auch [Script-Utility](#)).

Der Log-Viewer (c16\_lgbview.exe) befindet sich standardmäßig im CONZEPT 16-Serververzeichnis. Wird der CONZEPT 16-Server über die Installationsroutine installiert, werden zusätzlich alle \*.lgb-Dateien (Log Binary Files) mit diesem Programm verknüpft, sodass Log-Dateien direkt per Doppelklick betrachtet werden können.

Der Log-Viewer kann alternativ auch über die Kommandozeile gestartet werden. Die anzuzeigenden Logs werden dann durch Leerzeichen voneinander getrennt angegeben. Dabei ist eine Angabe von maximal zwei Log-Dateien möglich.

Bei Angabe der Option /help bzw. /? auf der Kommandozeile zeigt der Log-Viewer ein Meldungsfenster mit einer Kurzbeschreibung der Kommandozeile an.



Werden im Log-Viewer Log-Dateien aus einer neueren Version angezeigt, können neu hinzugefügte Meldungen nicht angezeigt werden. Beim Laden einer Log-Datei aus einer neueren Version wird beim Öffnen eine Warnung in der Statuszeile angezeigt.

## Kontakt

Typ	Datum	Uhrzeit	ID	Text
Init	01.12.2017	12:25:38		Process stopped
Data	01.12.2017	12:25:38		Database closed
Control	01.12.2017	12:25:38		Cache Information File saved (5374 segments)
Init	01.12.2017	12:25:37		Process stop requested
User	01.12.2017	12:25:37	13334	Logout START (DESKTOP-P4GGST8)
Control	01.12.2017	12:05:14		Cache loading completed (10 MB in 5s / read rate 2149 KB/s)
Data	01.12.2017	12:05:09		Database switched to read-write
HSB Master	01.12.2017	12:05:09		Disconnect from 10.1.4.38
HSB Master	01.12.2017	12:05:09		Loopback connect refused
HSB Master	01.12.2017	12:05:09		Connect to 10.1.4.38
HSB Master	01.12.2017	12:05:09		Connect to 192.168.0.1 failed (Socket connection failed)
User	01.12.2017	12:05:06	13334	Login START (DESKTOP-P4GGST8) / Windows 10 (64-Bit)
Control	01.12.2017	12:05:06		Cache Information File is valid
Data	01.12.2017	12:05:06		Database opened read only (2017-12-01 12:01:20) [22 MB size / 32 MB cache]
License	01.12.2017	12:05:05		License: CD100019MU/H, 100 user(s)
Init	01.12.2017	12:05:05		Process started (pid:5032) [5.8.08z-DBA-W64]
System	01.12.2017	12:05:05		System information: Windows 10 (64-Bit), 1 processor(s), 2047 MB RAM
Init	01.12.2017	12:01:21		Process stopped
Data	01.12.2017	12:01:20		Database closed

Zeile 1 / 41 CodeLibrary.lgb

Archiv-Zeitraum 01.12.2017 bis 01.12.2017

Der Log-Viewer kann über die Menüleiste sowie die Toolbar gesteuert werden. Mit [Öffnen] wird ein Dialog aufgerufen, über den die anzuseigende Log-Datei ausgewählt werden kann. Optional können in diesem Dialog bis zu zwei Log-Dateien zur Ansicht selektiert werden. Während des Ladens der Log-Dateien kann mit [Abbrechen] in der unteren linken Ecke der Ladevorgang vorzeitig abgebrochen werden. Falls im Verzeichnis der Log-Datei neben einem normalen Log (\*.lgb, \*.lgx) auch ein Archiv-Log (\*.lgbx, \*.lgux) liegt (siehe Architektur des Servers), wird das gesamte Log aus Standard-Log und Archiv-Log betrachtet. Falls der Log-Viewer über das Control-Center gestartet wurde, können alle Log-Dateien betrachtet werden, die über das Control-Center transferiert wurden. Diese werden dann entweder über das Dateimenü oder die aufklappbare Liste bei der [Öffnen]-Schaltfläche ausgewählt. Mit [Aktualisieren] werden die angezeigten Log-Dateien erneut eingelesen.

Die Log-Einträge werden standardmäßig nach Datum und Uhrzeit sortiert. Über einen Klick auf den Kopf der Datumsspalte kann zwischen einer aufsteigenden und absteigenden Sortierung gewechselt werden. Werden zwei verschiedene Log-Quellen angezeigt, werden diese in die korrekte zeitliche Abfolge gebracht, sodass wechselseitige Beziehungen zwischen den einzelnen Log-Einträgen betrachtet werden können. Zusätzlich können die Log-Einträge auch aufsteigend und absteigend nach dem Typ und dem enthaltenen Text sortiert werden. Dazu wird einfach der entsprechende Spaltenkopf angeklickt.



Der Zeitstempel der Eintragungen richtet sich nach der Systemzeit. Wird die Systemzeit geändert, wirkt sich das auf die neuen Eintragungen in den Protokolldateien aus. Es werden keine Eintragungen mit einem kleineren

## Kontakt

Zeitstempel als der letzte Eintrag in die Protokolldatei vorgenommen. Auf diese Weise wird verhindert, dass die Reihenfolge der Einträge verändert wird. Erfolgte versehentlich ein Eintrag mit einem Datum in der Zukunft, muss die Protokolldatei gelöscht werden, um wieder Eintragungen mit dem korrekten Zeitstempel zu erhalten.

Es werden maximal 500.000 (32-Bit-Variante) bzw. 2.000.000 (64-Bit-Variante) Einträge gleichzeitig angezeigt. Falls zwei verschiedene Log-Quellen angezeigt werden, verdoppelt sich diese Begrenzung. Wenn der nächste Abschnitt der Liste angezeigt werden soll, kann dies über das Menü Anzeige / Ältere Einträge gemacht

werden oder es wird die -Taste betätigt, wenn die letzte Zeile der Liste selektiert ist. Falls die Log-Einträge aufsteigend betrachtet werden, muss entsprechend die

-Taste betätigt werden, wenn die erste Zeile der Liste selektiert ist. Der gesamte Zeitraum, für den Log-Einträge vorhanden sind, wird in der Statusleiste am unteren rechten Rand angezeigt. Um zu einer bestimmten Stelle im Log zu springen, siehe Archiv. Werden mehr als 500.000 bzw. 2.000.000 Einträge angezeigt, ist es nur möglich nach Datum und Uhrzeit zu sortieren. Wird die 64-Bit-Variante des Log-Viewers verwendet, ist dies in der Titelzeile vermerkt.



Informationen zu den möglichen Log-Einträgen können dem Abschnitt Log-Einträge entnommen werden.

### Weitere Log-Datei Funktionen

#### Versenden

Log-Dateien können parallel zum Control-Center auch über den Log-Viewer versendet werden. Dazu wird der Menüpunkt Datei / Versenden aufgerufen. Anschließend öffnet der Log-Viewer den lokalen E-Mail Client mit dem vectorsoft Support-Empfänger und einer entsprechenden Betreffzeile. Die aktuell angezeigte Log-Datei wird dabei automatisch in die E-Mail eingefügt.

#### Eigenschaften

Über [Eigenschaften] in der Toolbar können die Eigenschaften der angezeigten Log-Dateien betrachtet werden. Wurden verschiedene Log-Dateien geladen, kann im Eigenschaftsfenster über die Auswahlliste das gewünschte Log ausgewählt werden. In den ersten Abschnitten wird die CONZEPT 16-Server Version angegeben, mit der als letztes die Log-Datei geschrieben wurde und einige Informationen zu der Log-Datei selbst. Im dritten Absatz der Eigenschaften werden nützliche Informationen zur Herkunft der Log-Datei angezeigt. Da der Name der Log-Datei dem Aliasnamen der Datenbank entspricht, wird unter Originalname der ursprüngliche Speicherort auf der Festplatte des Datenbankservers angezeigt. Zusätzlich wird der Rechnername, das Betriebssystem des Datenbankservers, die Anzahl der logischen Prozessoren und die Menge des physikalischen Speichers dargestellt. Die Werte sind von dem Server gespeichert, der zuletzt in die Log-Datei geschrieben hat.

#### Aktualisieren

Mit der Schaltfläche [Aktualisieren] werden die Log-Dateien erneut gelesen und angezeigt.

Wurde der Log-Viewer aus dem Control-Center gestartet, werden die

## Kontakt

Protokoll-Dateien erneut vom CONZEPT 16-Server abgeholt.

### Weitere Anzeigeoptionen

#### Datums-Änderungen

Zu jedem Log-Eintrag wird in der Spalte Datum das Datum angezeigt, an welchem der Log-Eintrag angelegt wurde. Um den Tageswechsel optisch hervorzuheben, kann unter Anzeige / Datums-Änderungen definiert werden, dass nur dann das Datum angezeigt wird, wenn sich das Datum zum vorhergehenden Eintrag auch geändert hat.

#### Zeilenselektion

Mit der Tastenkombination  /  kann ausgehend vom aktuell selektierten Eintrag zum vorhergehenden, beziehungsweise nächsten Eintrag mit der gleichen Fehlerklasse (Info, Warnung, Fehler) gesprungen werden.

Mit der Tastenkombination  +  /  +  wird ausgehend vom aktuell selektierten Eintrag auf den nächsten, beziehungsweise vorhergehenden Fehlereintrag positioniert. Ist bereits ein Fehlereintrag selektiert, passiert nichts. Auf den nächsten beziehungsweise vorhergehenden Fehlereintrag kann dann mit  /  positioniert werden.

#### Filter

Über den Menüpunkt Anzeige / Filter kann eine weitere Toolbar eingeblendet werden. Über diese ist es möglich einen Zeitraum anzugeben, welcher aus dem Log angezeigt werden soll. Dazu muss nur über die Datumseingabe bei [Start] und [Ende] ein Zeitraum angegeben und aktiviert werden. Wird ein neues Log geladen oder ein vorhandenes aktualisiert, wird dieser Zeitraum wieder zurückgesetzt. Über das Eingabefeld [Text] kann eine Zeichenkette eingegeben werden. Im Log werden dann nur Einträge angezeigt, welche diese Zeichenkette enthalten. Zwischen Groß- und Kleinschreibung wird dabei nicht unterschieden. Auf der rechten Seite können zusätzlich Log-Einträge bestimmten Typs und Klasse ausgeblendet werden. Dazu müssen die entsprechenden Schaltflächen der Filter-Toolbar aktiviert werden. Die Auswahl der auszublendenen Gruppen kann ebenfalls im Menü unter Anzeige / Ausblenden vorgenommen werden. Es ist so möglich, User-Einträge, Hot-Standby Einträge und sonstige Einträge, sowie Informationen, Warnungen, Fehler und Debug-Meldungen auszublenden.

## Kontakt

The screenshot shows the CONZEPT 16 Log-Viewer application window. The title bar reads "CONZEPT 16 Log-Viewer [CodeLibrary.lgb]". The menu bar includes "Datei", "Anzeige", and "?". Below the menu is a toolbar with icons for "Offnen" (Open), "Eigenschaften" (Properties), and "Aktualisieren" (Update). A search bar at the top right contains fields for "Start" (01.12.2017) and "Ende" (01.12.2017), a "Text" search field, and a "Ausblenden" (Hide) button. To the right of these are several small icons. The main area is a table with columns: "Typ", "Datum", "Uhrzeit", "ID", and "Text". The table lists log entries such as "Init", "Data", "Control", "User", and "HSB Master" events occurring on 01.12.2017 between 12:25:38 and 12:05:09. Some entries include IDs like 13334 and mention "Logout START", "Cache loading completed", and connection attempts to various IP addresses. The bottom of the table shows "Zeile 1 / 41" and "CodeLibrary.lgb". The footer indicates the archiving period from "01.12.2017 bis 01.12.2017".

### Archiv

Über den Menüpunkt Anzeige / Archiv kann eine weitere Toolbar eingeblendet werden. Über diese ist es möglich den Endzeitpunkt anzugeben, bis zu dem das Log angezeigt werden soll. Besonders bei Logs aus Standard- und Archiv-Log ist dies sinnvoll, um direkt zu einem bestimmten Zeitraum springen zu können. Der hier eingestellte Zeitraum wird auch verwendet, falls neue Log-Dateien geladen oder bereits angezeigte Log-Dateien aktualisiert werden.

### Hilfe (?)

Im Hilfe-Menü kann unter Hilfe die CONZEPT 16-Online-Dokumentation aufgerufen werden. Über den Menüpunkt Info wird der Informationsdialog des Log-Viewers geöffnet. Hier finden sich allgemeine Informationen über den CONZEPT 16 Log-Viewer, wie beispielsweise die Versionsnummer. Unter dem Registerreiter "Parameter" im Informationsdialog wird der Speicherort der verwendeten Log-Viewer Konfigurationsdatei angezeigt.

## Kontakt

### CONZEPT 16-Server - Log-Einträge

Beschreibung der Einträge in den Log-Dateien des CONZEPT 16-Servers In diesem Abschnitt werden die möglichen Inhalte der CONZEPT 16-Server Log-Dateien aufgeführt. Die Meldungen, die vom Datenbankserver protokolliert werden, sind in verschiedene Typen und Klassen unterteilt. Dies dient einerseits einem besseren Überblick bei der Ansicht. Zum anderen können so auch einfach alle Einträge eines bestimmten Typs oder Klasse ausgeblendet werden (zum Beispiel Login- und Logout-Daten, siehe auch Log-Viewer und Script-Utility). Die Typen werden folgendermaßen unterteilt.

**control** Meldungen zur Kontrolle der Datenbankzustände

**data** Meldungen zu Datenbankinhalten

**HSB master** Meldungen bezüglich Hot-Standby vom Server in der Master-Rolle **HSB slave**

Meldungen bezüglich Hot-Standby Server in der Slave-Rolle

**init** Meldungen beim Starten des Servers

**internal** Interne Fehlermeldungen

**I/O** Meldungen zum Lesen und Schreiben auf dem Speichermedium

**license** Meldungen zur Lizenzierung

**request** Meldungen zu Benutzeranfragen an den Server

**system** Meldungen zum Betriebssystem

**user** Meldungen zu Benutzeraktionen wie zum Beispiel Login und Logout

Jede Klasse hat ein zugeordnetes Kürzel. Über dieses kann beim Betrachten eines Logs über das Script-Utility die Klasse ermittelt werden. Die Aufteilung der Klassen entspricht dem folgenden Schema:

#### Klasse Kürzel Beschreibung

**Informationen** i Informationsmeldungen. Kein fehlerhaftes Verhalten.

**Warnungen** w Warnungsmeldungen, die den Betrieb des Servers nicht beeinträchtigen, jedoch vom Administrator aufmerksam verfolgt werden sollten.

**Fehler** e Fehlermeldungen, die unbedingt beachtet und behoben werden müssen.

**Debug-Meldungen** d Meldungen, die der Datenbankserver zusätzlich ausgibt, wenn interne Debug-Schalter aktiv sind.

Ist die automatische E-Mail-Benachrichtigung des Servers eingerichtet, wird bei Meldungen der Klassen "w" und "e" eine E-Mail versendet.

Die Protokolleinträge gliedern sich in folgende Funktionsabschnitte:

- Allgemeine Fehler
- Datenbankfehler
- Starten und Stoppen des Servers
- Server Ereignisse
- Server Ereignisse (Rollback)
- Server Ereignisse (Debug)
- Server Hauptprozess
- Server Managerprozess

## Kontakt

- Server Datenraumtabelle
- Server Lizenzverarbeitung
- Internetbasierte Lizenzen
- Server Datenbankprozess
- Server Speichermanagement
- Server Datenzugriff
- Server Anfragefehler
- Server Ein-/Ausgabefehler
- Server Hot-Standby
- SOA-Service
- Laufwerkstreiber
- Datenbankreplikation

Einige Protokolleinträge geben einen zusätzlichen Fehlerwert aus, um den Fehler genauer zu spezifizieren. Diese Einträge gliedern sich in folgende Bereiche:

- Dateizugriffs-Fehler
- Socket-Fehler
- Kommunikationsfehler mit einem Lizenzserver
- Unbestimmte Fehlerwerte

Allgemeine Fehler:

- Alert mail was send to ...

Eine Alert-E-Mail wurde an die angegebene Adresse versendet. Dies ist nur ein informativer Eintrag.

- CRC Error

Beim CRC-Prüfsummenvergleich ist eine Abweichung festgestellt worden. Die Kommunikation läuft nicht fehlerfrei. Fehler können auftreten bei Inkompatibilitäten, Hardware- oder Software-Fehlern.

- Failed to send alert mail (code ...)

Die Fehlermeldung konnte nicht per E-Mail versendet werden. Die Einstellungen der automatischen Mail-Benachrichtigung müssen überprüft werden.

- Failed to send alert mail to ... (code ...)

Die Fehlermeldung konnte nicht per E-Mail versendet werden. Die Adresse zu der nicht gesendet werden konnte und der Fehlerwert werden angezeigt. Einstellungen der automatischen Mail-Benachrichtigung und des E-Mails-Servers müssen überprüft werden.

- no error code

Es konnte kein Fehlerwert ermittelt werden.

- Out of memory

Der Prozess konnte den notwendigen Hauptspeicher nicht anfordern. Möglicherweise steht nicht genug Hauptspeicher zur Verfügung oder der Speicher konnte aus anderen Gründen nicht angefordert werden.

## Kontakt

- **Thread start failed**

Ein Thread konnte nicht gestartet werden. Möglicherweise ist ein Ressourcenlimit des Betriebssystem erreicht. Der Rechner sollte neu gestartet werden.

- **Timeout**

Es ist eine Zeitüberschreitung eingetreten. Möglicherweise ist der Rechner stark ausgelastet.

## Datenbankfehler

- **Database in backup mode**

Die Datenbank befindet sich im Backup-Modus. Der Backupmodus kann über den Befehl DbaControl() oder das Script-Utility gestartet und gestoppt werden.

- **Database in standby mode**

Die Datenbank konnte nicht geöffnet werden, da sie im Standby-Modus ist.

- **Database is locked (...)**

Die Datenbank ist gesperrt. Die Sperre kann explizit eingerichtet oder durch Operationen auf der Datenbank implizit definiert worden sein. Die Art der Sperre wird in Klammern angegeben (siehe Fehler beim Öffnen der Datenbank).

- **Database name conflict**

Es gibt bereits eine Datenbank mit dem gleichen symbolischen Namen oder mit dem gleichen Dateinamen. Die Eintragungen der Datenbanken müssen überprüft werden (siehe Konfiguration des Servers).

- **Database open denied**

Die Datenbank konnte aufgrund des Datenbankstatus nicht geöffnet werden. Dieser Fehler kann auftreten, wenn sich die Datenbank in Synchronisation oder im Rollback befindet.

- **Database open failed (...)**

Beim Öffnen der Datenbank ist ein Fehler aufgetreten. Der Fehler wird in Klammern angegeben. Hier können unterschiedliche Fehler im Bereich des Dateisystems vorliegen, oder dass eine nicht unterstützte Datenbankversion versucht wurde zu öffnen.

- **Database service initialisation failed**

Der Datenbankprozess konnte nicht initialisiert werden. Möglicherweise lässt sich das Problem durch einen Neustart des Systems beseitigen.

- **Database unknown**

Zu dem angegebenen symbolischen Namen existiert keine Datenbank. Der Eintrag der Datenbank muss überprüft werden (vgl. Konfiguration des Servers).

- **Invalid password**

Es wurde ein falsches Kennwort angegeben.

## Kontakt

- **Log file open failed**

Die Log-Datei konnte nicht geöffnet werden. Hier können Probleme im Bereich der Dateiverwaltung vorliegen.

- **User limit reached**

Die maximale Anzahl der Benutzer ist erreicht. Die Anzahl der in der Lizenz angegebenen Benutzer wurde überschritten oder der Datenbankcache ist für die Anzahl der Benutzer nicht mehr ausreichend.

- **Wrong database version**

Die Version der zu öffnende Datenbank wird nicht vom Server oder von der verwendeten Lizenz unterstützt. Datenbank Versionen kleiner als 4.7 und größer als die Releasenummer des Service können nicht geöffnet werden.

### Starten und Stoppen des Servers

- **Communication port is already in use**

Der Client-Port des Datenbankservers (4722) ist bereits belegt. Dies ist beispielsweise dann der Fall, wenn bereits eine ältere Version (1.8/5.0) des Datenbankservers gestartet ist.

- **Detached process can't be started in a terminal session**

Der Datenbankserver kann im Detached-Mode nicht innerhalb einer Terminalsitzung gestartet werden.

- **Operating system not supported**

Der Prozess kann unter diesem Betriebssystem nicht gestartet werden. Entweder ist das Betriebssystem noch nicht für CONZEPT 16 freigegeben oder die Voraussetzungen für den Einsatz einer bestimmten Lizenz sind nicht gegeben (siehe Lizenztypen).

- **Process authorisation failed**

Der Prozess konnte nicht gestartet werden, da keine gültige Startauthorisierung vorhanden ist. Die Eintragungen bei dem Dienst des Systems müssen überprüft werden.

- **Process initialisation failed**

Die Initialisierung des Prozesses ist fehlgeschlagen. Möglicherweise lässt sich das Problem durch einen Neustart des Systems beseitigen.

- **Process started (...) [...]**

Ein Prozess wurde gestartet. In runden Klammern wird die Prozess-ID und in eckigen Klammern der Typ, die Version und die Startart (service oder detached) des Prozesses angegeben. Die Startart taucht nur beim Service-Prozess auf, da Manager- und Datenbankprozess generell vom Service-Prozess im Detached-Mode gestartet werden. Dies ist selbst dann der Fall, wenn der Service-Prozess als Dienst eingerichtet ist. Vor dem Start des Prozesses werden Informationen über das System (Betriebssystem, Anzahl der logischen Prozessoren und Menge des physikalischen Speichers) ausgegeben.

## Kontakt

- **Process start failed [...]**

Der Prozess konnte nicht gestartet werden. In Klammern wird der Typ und die Version des Prozesses angegeben. Die Log-Einträge vor diesem Eintrag müssen ausgewertet werden.

- **Process stopped**

Der Prozess wurde angehalten.

- **Process term delay (...)**

Das Beenden des Prozesses wird um die in Klammern angegebene Zeit verzögert.

Siehe auch Konfiguration des Servers.

- **Program file damaged**

Der Prozess konnte nicht gestartet werden, da die Programmdatei beschädigt ist. In diesem Fall muss der Programmstand noch einmal installiert werden. Möglich sind auch Probleme beim Dateisystem.

- **Server is already started**

Der Datenbankserver ist bereits gestartet. Der CONZEPT 16-Server kann auf einem Rechner nur einmal gestartet werden, auch dann wenn der Server in verschiedenen Benutzerkontexten läuft.

- **Shared memory access failed**

Der Zugriff auf den gemeinsamen Speicherbereich des Datenbankservers ist fehlgeschlagen. Möglicherweise lässt sich das Problem durch einen Neustart des Systems beseitigen.

## Server Ereignisse

- **Alert mail test**

Der Test der automatischen E-Mail-Benachrichtigung wurde ausgelöst. Der Test erzeugt eine E-Mail. Der Empfang der E-Mail muss ebenfalls überprüft werden.

- **Backup event completed**

Ein Sicherungereignis wurde beendet.

- **Backup event started (...)**

Ein Sicherungereignis wurde gestartet. Die voraussichtliche Dauer des Ereignisses wird in Klammern angegeben. Der Eintrag erfolgt nur, wenn die Datenbank während des Backup-Ereignisses geöffnet ist oder wird.

- **Client communication: CRC disabled**

Die Absicherung der Kommunikation zwischen den Clients und dem Server über eine CRC-Prüfsumme wurde deaktiviert (Datentransfer mit CRC).

- **Client communication: CRC enabled**

Die Absicherung der Kommunikation zwischen den Clients und dem Server über eine CRC-Prüfsumme wurde aktiviert (Datentransfer mit CRC).

## Kontakt

- **Database closed**  
Die Datenbank wurde geschlossen.
- **Database opened (...) [...]**  
Die Datenbank wurde geöffnet. Der Zeitpunkt der letzten Änderung der Datenbankdatei (.ca1) wird in runden Klammern, die Größe der Datenbank und der verwendete Cache in eckigen Klammern angegeben.
- **Database opened read only (...)**  
Die Datenbank wurde im Nur-Lesen-Modus geöffnet. Der Zeitpunkt der letzten Änderung der Datenbankdatei (.ca1) wird in Klammern angegeben. Alle Änderungen in der Datenbank erfolgen nur im Cache. Nach dem Schließen der Datenbank werden alle Änderungen verworfen.
- **Database rollback unlocked**  
Der Rollback wurde für die Datenbank freigegeben (Freigabe für Rollback).
- **Database switched to read-only**  
Der Zugriffsmodus wurde auf "Nur-Lesen" geändert. Dies ist notwendig bei verschiedenen Datenbankoperationen (zum Beispiel Backup, Diagnose usw.).
- **Database switched to read-write**  
Der Zugriffsmodus wurde auf "Lesen-Schreiben" geändert.
- **Database synchronisation unlocked**  
Die Synchronisation wurde für die Datenbank freigegeben (Freigabe für Synchronisation).
- **Database user login disabled**  
In der Datenbank wurde die Anmeldesperre gesetzt (Login sperren).
- **Database user login enabled**  
In der Datenbank wurde die Anmeldesperre entfernt (Login freigeben).
- **Database user login unlocked**  
Die Aktivierung wurde für die Datenbank freigegeben (Freigabe für Login).
- **Failed to send alert mail (code ...)**  
Die automatische E-Mail-Benachrichtigung konnte nicht durchgeführt werden. Der Fehlerwert wird in Klammern angegeben (siehe MailClose()).
- **HSB master server switched to primary system**  
Es wurde der Wechsel zum Primärsystem durchgeführt (Wechsel zu Primärsystem).
- **HSB master server switched to secondary system**  
Es wurde der Wechsel zum Sekundärsystem durchgeführt (Wechsel zu Sekundärsystem).
- **Login ... (...) / ...**  
Ein Benutzer hat sich erfolgreich an der Datenbank angemeldet. Der Name des

## Kontakt

Benutzers, des Client-Computers und des verwendeten Betriebssystems werden ebenfalls angegeben.

- **Login failed...**

Die Anmeldung eines Benutzers wurde abgewiesen. Der Name des Benutzers und des Client-Computers werden ebenfalls angegeben. Möglicherweise wurden die falschen Anmelddaten verwendet.

- **Logout...**

Ein Benutzer hat sich abgemeldet. Der Name des Benutzers und des Client-Computers werden ebenfalls angegeben.

- **Forced user logoff ...**

Ein Benutzer wurde zwangsweise (Menübefehl des Standard-Clients bzw. UserClear()) abgemeldet. Der Name des Benutzers und des Client-Computers werden ebenfalls angegeben.

- **Procedure runtime error ...**

Bei der Durchführung einer Funktion beim Server mit der Anweisung RmtCall() ist ein Laufzeitfehler aufgetreten. Die Nummer des Laufzeitfehlers, die aufgerufene Funktion, die Zeile, in der der Fehler aufgetreten ist und der Klartext der Fehlermeldung werden in dem Eintrag ausgegeben.

- **VSS Backup event completed [(timeout)]**

Ein Sicherungsereignis wurde über den Volumenschattenkopiedienst beendet. Wenn das Sicherungsereignis länger als 5 Minuten benötigt, wird es abgebrochen. In diesem Fall steht in Klammern "timeout" hinter dem Log-Eintrag.

- **VSS Backup event started**

Ein Sicherungsereignis wurde über den Volumenschattenkopiedienst gestartet. Der Eintrag erfolgt nur, wenn die Datenbank während des Backup-Ereignisses geöffnet ist oder wird.

- **VSS Library load failed**

Die DLL für den Volumenschattenkopiedienst (c16\_serv\_vss\_\*.dll) wurde nicht gefunden oder konnte nicht initialisiert werden.

## Server Ereignisse (Rollback)

- **... blocks restored from transaction log file TL?**

Aus der Transaktionslogdatei wurden ... Blöcke in die Datenbank übertragen. Zusätzlich wird angezeigt, aus welchem Zeitraum die übertragenen Blöcke stammen.

- **No valid transaction log files found (...)**

Es wurden keine verwendbaren Transaktionslogdateien für den Rollback gefunden. Der Grund wird in Klammern angegeben:

- Die Transaktions-Logdateien konnten nicht geöffnet werden (open failed).

## Kontakt

- Die Transaktions-Logdateien konnten geöffnet werden, jedoch ist der Header korrupt (corrupt header ).
  - Die Transaktions-Logdateien sind gültig, jedoch passt die Datenbank nicht zu den Logdateien (database mismatch).
  - Die Transaktions-Logdateien passen zur Datenbank, enthalten jedoch nicht die richtige Seriennummer (wrong serial number).
- **Transaction log file ID error**  
Beim Übertragen der Daten aus einer Transaktionslogdatei in die Datenbank wurde ein ungültiger Segmenteintrag gefunden.
- **Transaction log file read fault**  
Beim Lesen der Transaktionslogdatei ist ein Fehler aufgetreten. Die Fehler sind im Bereich des Dateisystems zu suchen.
- **Transaction log file TL? has no valid blocks**  
Die Transaktionslogdatei ist nicht verwendbar, da sie entweder beschädigt oder unvollständig ist.
- **Transaction Rollback finished Das Rollback wurde beendet.**
- **Transaction Rollback started (...)**  
Ein Rollback wurde gestartet. Der Grund wird in Klammern angegeben. Entweder wurde eine nicht eingespielte Transaktionslog-Datei gefunden, oder die Datenbank wurde zuvor nicht ordnungsgemäß geschlossen.

## Server Ereignisse (Debug)

- **User long request OP ... time ...**  
Eine Anfrage benötigt länger als drei Sekunden zur Verarbeitung. Es wird der Operationscode und die Dauer der Verarbeitung in Sekunden ausgegeben. Die Dauer der Verarbeitung ist immer in Verbindung mit der Operation zu sehen. Es gibt Operationen, die immer länger als drei Sekunden dauern.
- **User wait on ID ... tree ... seg ...**  
Bei einer Anfrage ist ein Wartezustand von mehr als 2 Sekunden Dauer eingetreten. Es wird die ID des Benutzer ausgegeben, der das Segment gesperrt hält. Zusätzlich wird die Nummer des Baums angezeigt, zu dem das gesperrte Segment gehört. Die Einträge dienen zur Überprüfung der eigenen Programmierung im Bereich der Transaktionen.

## Server Hauptprozess

- **Connect to manager process failed (...)**  
Es konnte keine Verbindung zum Manager-Prozess aufgenommen werden. Der Fehler wird in Klammern angegeben. Diese Meldung kann auch in den Datenbankprozessen auftreten. Der FehlerText gibt Hinweise auf die Ursache des Problems.

## Kontakt

- Manager process launched

Der Start des Managerprozesses wurde eingeleitet.

- Service started [...]

Der Serviceprozess wurde gestartet. In eckigen Klammern ist die Version angegeben.

- Service start failed

Beim Starten des Serviceprozesses ist ein Fehler aufgetreten. Dies ist lediglich das Resultat eines zuvor aufgetretenen Fehlers.

- Service start requested

Der Serviceprozess soll gestartet werden.

- Service stopped

Der Serviceprozess wurde angehalten.

- Service stop requested

Der Serviceprozess soll angehalten werden.

- Service stop requested (shutdown)

Der Serviceprozess soll aufgrund des Herunterfahrens des Systems angehalten werden.

- Start of manager process failed (...)

Beim Start des Managerprozesses ist ein Fehler aufgetreten. Der Fehlertext wird in Klammern angegeben und gibt Hinweise für die Ursache.

## Server Managerprozess

- Service process release mismatch

Die Version des Serviceprozesses ist nicht kompatibel mit dem Release des Managerprozesses. Der Programmstand sollte noch einmal installiert werden.

- Start of communication services failed

Die Verbindungsdienste des Managerprozesses konnten nicht gestartet werden. Möglicherweise liegt ein Portkonflikt vor (vgl. Architektur des Servers).

- Start of database process failed [...] (...)

Beim Starten des Datenbankprozesses ist ein Fehler aufgetreten. Der Name der zu öffnenden Datenbank und der Fehler werden in Klammern angegeben. Der Fehlertext gibt Hinweise auf das Problem.

- Start of database process failed (timeout)

Beim Starten des Datenbank-Prozesses ist eine Zeitüberschreitung aufgetreten. Möglicherweise ist das System überlastet.

## Server Datenraumtabelle

## Kontakt

- **Database table read fault**

Beim Lesen der Datenraumtabelle ist es zu einem Fehler gekommen. Die Datei (c16\_serv.ars) ist beschädigt oder es liegt ein Problem beim Dateisystem vor.

- **Database table damaged**

Beim Einlesen der Datenraumtabelle sind ungültige Inhalte festgestellt worden. Die Datei (c16\_serv.ars) muss überprüft werden. Möglicherweise gibt es auch ein Problem beim Dateisystem.

- **Database table write fault (...)**

Beim Schreiben der Datenraumtabelle ist es zu einem Fehler gekommen. Der Fehler des Betriebssystems wird in Klammern angegeben.

## Server Lizenzverarbeitung

- **Client type is not supported by this license**

Der Client-Typ wird von der Lizenz nicht unterstützt. Nähere Informationen siehe Lizenztypen.

- **Client upgrade missing**

Das Client-Release kann mit der Lizenz-Version nicht verwendet werden. Es sollte die aktuelle Lizenzdatei im Kundenbereich von vectorsoft heruntergeladen werden.

- **Communication error**

Fehler bei der Kommunikation mit dem Server.

- **Database or server in evaluation mode**

Die Datenbank oder der Server sind im Evaluierungsmodus. Zum Betrieb einer Evaluierungsdatenbank muss sich der Server ebenfalls im Evaluierungsmodus befinden.

- **Different Dongle detected**

Anstelle eines korrekten CONZEPT 16-Dongles wurde ein anderer Dongle gefunden. Sind Dongle von anderen Anbietern vorhanden, sollten diese entfernt werden.

- **Dongle driver not installed**

Der Dongle-Treiber konnte nicht gefunden werden. Es muss sichergestellt sein, dass der Treiber korrekt eingerichtet wurde.

- **Dongle not found**

Der Dongle wurde nicht gefunden. Ohne Dongle hat der Server keine gültige Lizenz, somit können keine Datenbanken geöffnet werden. Die Fehlermeldung tritt auch beim Sekundärserver in einem Hot-Standby-System auf. Der Sekundärserver benötigt in diesem Fall keinen Dongle. In den Einstellungen sollte der Eintrag "Hot-Standby-Betrieb ohne Dongle" aktiviert werden, damit keine Fehlerprotokolle per E-Mail versendet werden. Dieser Fehler wird nur einmal in 24 Stunden protokolliert.

## Kontakt

- **Dongle port is busy**

Der Dongle konnte auch nach mehreren Versuchen nicht gelesen werden, weil die Schnittstelle beschäftigt ist. Andere Geräte an der Schnittstelle sollten entfernt werden.

- **Dongle read error**

Beim Lesen des Dongles ist ein Fehler aufgetreten. Die Funktion des Dongles sollte an einem anderen System mit dem Programm c16\_licinfo.exe überprüft werden. Dazu muss auf diesem System der Dongle-Treiber installiert werden.

- **Dongle read successfully**

Der Dongle wurde korrekt gelesen.

- **Dongle type not supported**

Der Typ des Dongles wird nicht unterstützt. Die Funktion des Dongles sollte an einem anderen System mit dem Programm c16\_licinfo.exe überprüft werden. Dazu muss auf diesem System der Dongle-Treiber installiert werden.

- **Dongle version not supported**

Die Version des Dongles wird nicht unterstützt. Die Funktion des Dongles sollte an einem anderen System mit dem Programm c16\_licinfo.exe überprüft werden. Dazu muss auf diesem System der Dongle-Treiber installiert werden.

- **Evaluation License expired on ...**

Die Evaluierungslizenz ist abgelaufen.

- **Hot standby license (...) has expired (...)**

Die Lizenz für den Hot-Standby-Betrieb ist abgelaufen.

- **Hot standby license (...) is not valid**

Die angegebene Lizenz für den Hot-Standby-Betrieb ist nicht gültig. Es sollte die aktuelle Lizenzdatei bei vectorsoft heruntergeladen werden.

- **Hot standby license (...) is valid. License expires on ...**

Die angegebene temporäre Lizenz für den Hot-Standby-Betrieb endet an dem angegebenen Zeitpunkt.

- **Invalid or damaged license file**

Die Lizenzdatei ist ungültig oder beschädigt. Die Lizenzdatei muss ersetzt werden.

Möglicherweise gibt es auch ein Problem beim Dateisystem.

- **License error occured**

Lizenznummer ist ungültig.

- **License (...) expired on ...**

Der CONZEPT 16-Server läuft mit einer temporären Lizenz. Die Lizenz ist abgelaufen.

- **License ... expires on ...**

Der CONZEPT 16-Server läuft mit einer temporären Lizenz. Die Lizenz endet an dem angegebenen Zeitpunkt.

## Kontakt

- License file damaged

Die Lizenzdatei ist beschädigt. Es sollte die aktuelle Lizenzdatei im Kundenbereich von vectorsoft heruntergeladen werden.

- License file incomplete

Die Lizenzdatei ist unvollständig. Es sollte die aktuelle Lizenzdatei im Kundenbereich von vectorsoft heruntergeladen werden.

- License file mismatch

Die Lizenzdatei passt nicht zu der Lizenz. Es sollte die aktuelle Lizenzdatei im Kundenbereich von vectorsoft heruntergeladen werden.

- License file not found

Die Lizenzdatei c16.lic wurde nicht gefunden. Die Datei kann im Kundenbereich von vectorsoft heruntergeladen werden.

- License file read successfully (...)

Die Lizenzdatei (c16.lic) wurde korrekt eingelesen. In Klammern wird das Datum und die Uhrzeit der Erstellung der Lizenzdatei ausgegeben.

- License is valid (...)

Die verwendete Lizenz ist gültig. In Klammern wird die Lizenznummer, die maximale Anzahl der Benutzer und das maximal unterstützte Release angegeben.

- License number not found

In der Lizenzdatei wurde die Lizenznummer des Dongles nicht gefunden. Es muss die aktuelle Lizenzdatei verwendet werden. Die Datei kann im Kundenbereich von vectorsoft heruntergeladen werden.

- License upgrade missing

Die Lizenz ist nicht für die verwendete Version freigegeben. Es muss sichergestellt sein, dass die aktuelle Lizenzdatei verwendet wird. Die Datei kann im Kundenbereich von vectorsoft heruntergeladen werden.

- User login denied (license limit reached: ...)

Der Benutzer konnte nicht angemeldet, weil die maximale Benutzeranzahl des Servers erreicht wurde. Die maximale Anzahl der Benutzer des Servers wird hinter dem Doppelpunkt angegeben. Bei dem angezeigten Wert ist die Anzahl der reservierten Benutzer bereits abgezogen.

- Missing Dongle. License expires on ...

Der Dongle konnte im laufenden Betrieb nicht gelesen werden. Die Lizenz ist noch bis zu dem angegebenen Zeitpunkt gültig. Ab diesem Zeitpunkt können sich keine Anwender mehr an Datenbanken anmelden. Die Fehlermeldung wird nur alle 24 Stunden protokolliert.

- Missing Dongle. License has expired

Der Dongle konnte im laufenden Betrieb nicht gelesen werden. Der Zeitpunkt, bis zu dem sich noch neue Benutzer an den Datenbanken anmelden konnten ist

## Kontakt

überschritten.

- **More than one dongle detected**

Es ist mehr als ein Dongle für den CONZEPT 16-Server vorhanden. Auf einem System kann nur eine Server-Lizenz verwendet werden. Es darf nur eine CONZEPT 16-Lizenz vorhanden sein. Andere Dongle sollten von dem System entfernt werden.

- **Operating system not supported by this license**

Das verwendete Betriebssystem wird nicht von der Lizenz unterstützt. Die Voraussetzungen für den Einsatz der Lizenz müssen überprüft werden.

- **Operating system on HSB slave not supported by this license**

Der Slave-Server kann mit der Lizenz des Master-Servers nicht betrieben werden, weil die Voraussetzungen zum Betrieb der Lizenz nicht erfüllt sind (siehe Lizenztypen). Hot-Standby ist nicht möglich.

- **Server release not supported**

Das Server-Release wird nicht unterstützt. Der Server muss mindestens die Version des jeweiligen Clients haben.

- **Transmission error while license checking** Übertragungsfehler bei der Kommunikation mit dem Server.

- **Unknown dongle error**

Beim Lesen des Dongles kam es zu einem unbekannten Fehler. Die Funktion des Dongles sollte an einem anderen System mit dem Programm c16\_licinfo.exe überprüft werden. Dazu muss auf diesem System der Dongle-Treiber installiert werden.

## Internetbasierte Lizenzen

- **Communication with License Server ... failed (...)**

Es wurde eine Kommunikationsverbindung mit dem angegebenen Lizenzserver aufgebaut, die aber unterbrochen wurde. Eine genauere Beschreibung des Fehlers wird in Klammern angegeben (siehe unten).

- **Connect to any License Server failed (...)**

Zu keinem der Lizenzserver konnte eine Verbindung hergestellt werden. Eine genauere Beschreibung des Fehlers wird in Klammern angegeben (siehe unten). Der Fehler wird nur einmal protokolliert, kann wieder eine Verbindung hergestellt werden, erfolgt der Eintrag Successful connect to License Server ....

- **Connect to License Server ... failed (...)**

Der angegebene Lizenzserver konnte erreicht, aber keine Kommunikationsverbindung aufgebaut werden. Eine genauere Beschreibung des Fehlers wird in Klammern angegeben (siehe unten).

- **License File updated successfully (...)**

Die Lizenzdatei (c16.lic) wurde erfolgreich auf den aktuellen Stand gebracht. In

## Kontakt

**Klammern wird der Produktionszeitpunkt der Lizenzdatei angegeben.**

- License File update failed (...)

**Die Lizenzdatei konnte nicht geschrieben werden. In Klammern steht eine nähere Erläuterung, welcher Fehler aufgetreten ist.**

- License File update query failed (...)

**Es ist ein Fehler bei der Abfrage beim Update der Lizenzdatei aufgetreten. Der Fehler wird in Klammern angegeben.**

- License Identity expires on ...

**Dieser Eintrag erfolgt nach einem Fehler bei einer Lizenzabfrage. Sie gibt an, bis wann die Lizenz noch gültig ist. Das Datum und die Uhrzeit sind in die lokale Zeit umgerechnet.**

- License Identity file is valid (...)

**Die Lizenz-Identitätsdatei ist gültig. Die Nummer der Lizenz wird in Klammern angegeben.**

- License Identity file read failed (...)

**Die Lizenz-Identitätsdatei konnte nicht gelesen werden. Eine genauere Beschreibung des Fehlers wird in Klammern angegeben (siehe unten).**

- License Identity file update failed (...)

**Die Lizenz-Identitätsdatei konnte nicht aktualisiert werden. Als Folge davon kann der CONZEPT 16-Server nicht verwendet werden. Eine genauere Beschreibung des Fehlers wird in Klammern angegeben (siehe unten).**

- License Identity has expired (...)

**Die Gültigkeit der Lizenz-Identität ist abgelaufen. Das Ende-Datum wird in Klammern mit ausgegeben. Das Datum und die Uhrzeit sind in die lokale Zeit umgerechnet.**

- License Identity is valid thru ...

**Die Lizenz-Identität ist gültig. Der Ablaufzeitpunkt der Gültigkeit wird mit angegeben. Der Eintrag erfolgt nur, wenn sich die Gültigkeit um mehr als 12 Stunden verändert hat. Das Datum und die Uhrzeit sind in die lokale Zeit umgerechnet.**

- License Identity not useable

**Die Lizenz-Identität kann nicht verwendet werden. Der CONZEPT 16-Server öffnet keine Datenbank. Verschiedene Ursachen können dazu führen, dass die Lizenz-Identität nicht mehr verwendet werden kann, entsprechende Fehlermeldungen werden vorher protokolliert.**

- License Identity verification failed (...)

**Die Antwort auf eine Lizenzabfrage kann nicht verarbeitet werden. Eine genauere Beschreibung des Fehlers wird in Klammern angegeben (siehe unten).**

- Old License Key used

**Ein alter Lizenzschlüssel wurde verwendet. Möglicherweise wurde eine ältere**

## Kontakt

Lizenz-Identitätsdatei verwendet. Tritt der Eintrag häufig auf, könnte eine Mehrfachnutzung der Lizenz vorliegen.

- Successful connect to License Server ...

Es wurde eine Verbindung zu einem der Lizenzserver aufgebaut. Der Name des Lizenzservers wird mit angegeben.

- System configuration change detected

Eine Änderung der Systemkonfiguration (DNS- oder NetBIOS-Name, Anzahl der logischen Prozessoren, Arbeitsspeicher oder Betriebssystemversion) wurde vorgenommen.

- System time adjustment failed

Die Systemzeit der lokalen Maschine weicht von der Systemzeit der Lizenzserver um mehr als eine Stunde ab. Die Systemzeit konnte nicht angepasst werden. Die Lizenz kann nicht mehr verwendet werden, wenn die Systemzeit um mehr als 24 Stunden abweicht.

- System time sucessfully adjusted

Die Systemzeit der lokalen Maschine weicht von der Systemzeit der Lizenzserver um mehr als eine Stunde ab. Die Systemzeit wurde entsprechend angepasst.

## Server Datenbankprozess

- Manager process release mismatch

Das Release des Managerprozesses ist nicht identisch mit dem Release des Datenbankprozesses. Der Programmstand sollte nochmal installiert werden.

- Database cache allocation failed

Der Datenbankpuffer konnte nicht eingerichtet werden. Meist ist die Ursache, dass zu wenig Speicher frei ist.

- Database cache extension failed (current size: ... MB)

Der Datenbankcache konnte nicht auf die eingestellte maximale Größe erweitert werden. Der Cache hat jetzt die angegebene Größe und wird danach auch nicht mehr erweitert.

- Database cache size limited to ... MB

Es wurde mehr Datenbankcache angegeben, als physikalischer Speicher zur Verfügung steht. Die maximale Cachegröße wurde auf den angegebenen Wert reduziert.

- Database is in synchronisation state

Die Datenbank kann nicht geöffnet werden, da sie im Zustand der Synchronisation ist.

- Database open is locked

Die Datenbank kann nicht geöffnet werden, da die Aktivierungssperre gesetzt ist.

## Kontakt

- **Database rollback is locked**  
Die Datenbank kann nicht geöffnet werden, da die Rollbacksperre gesetzt ist.
- **Database TMP file open failed**  
Die temporäre Auslagerungsdatei (<Datenbankname>.tmp) konnte nicht geöffnet werden. Das Dateisystem und die Rechte des Servers sollten überprüft werden.
- **Database TRS file open failed (...)**  
Die temporäre Transaktionsdatei (<Datenbankname>.trs) konnte nicht geöffnet werden. Der Fehler ist in Klammern angegeben. Das Dateisystem und die Rechte des Servers sollten überprüft werden.
- **License ..., ...**  
In der Log-Datei des Datenbankprozesses wird die verwendete Lizenz und die maximale Anzahl der Benutzer eingetragen. Der Eintrag dient nur der Information.
- **License ... (HSB database), ..., expires on ...**  
Dieser Eintrag erfolgt nur, wenn die Datenbank in einem Hot-Standby-System auf dem Sekundärserver als Master geöffnet wird. Es wird die Lizenznummer, die maximale Anzahl der Benutzer und der Zeitpunkt, zu dem die Lizenz des Sekundärservers endet ausgegeben. Der Eintrag dient nur zur Information.
- **Startup arguments incomplete**  
Beim Start des Datenbankprozesses fehlen Startparameter. Möglicherweise wurden bei einem Update nicht alle Programmdateien des Servers ersetzt.

## Server Speichermanagement

Die "Free table" ist eine Tabelle mit allen freien und belegten Segmenten in der Datenbank. Die Tabelle wird aus der Datenbank geladen und verbleibt im Speicher bis die Datenbank wieder geschlossen wird. Zum Speichern der Tabelle in der Datenbank wird diese komprimiert und in der Datenbank abgelegt. Wird ein Fehler gefunden, wird die Tabelle reorganisiert. In dieser Zeit kann die Datenbank nicht geschlossen werden.

- **Free table: CRC error**  
Es wurde ein Fehler innerhalb einer Tabelle gefunden.
- **Free table: invalid segment link**  
Die Verknüpfung zu einem Segment ist nicht korrekt.
- **Free table: invalid segment type**  
Innerhalb der freien Segmente befindet sich ein Segment falschen Typs.
- **Free table: no start segment**  
Die Tabelle der freien Segmente besitzt kein Startsegment.

## Kontakt

- **Free table: ... orphaned segment(s) recovered**

Von der Diagnose wurden ein oder mehrere leere Segmente, die nicht in der Tabelle der freien Segmente standen festgestellt und dieser Tabelle hinzugefügt. Die Anzahl der Segmente wird nach dem Doppelpunkt angegeben.

- **Free table: rebuild failed - data area is read-only**

Die Tabelle der freien Segmente konnte nicht neu aufgebaut werden, da die Datenbank sich im Nur-Lesen-Modus befindet.

- **Free table: serial number mismatch**

Es wurde ein Fehler innerhalb der Nummerierung der Segmente gefunden.

## Server Datenzugriff

Der Datensatzzugriff erfolgt entweder in den Datenraum (Bereich Data) oder durch eine Anfrage eines Clients (Bereich Request). Kommt es in diesen Bereichen zu Fehlermeldungen, sollte zunächst eine Diagnose durchgeführt werden. Die Diagnose gibt dann bereits Informationen über die betroffenen Bereiche und das Ausmaß des Problems. Nach dem Anfertigen einer Datensicherung kann anschließend je nach Problemfall eine Diagnose mit Recover, ein Schlüsselreorganisation usw. durchgeführt werden.

- **Prime counter overflow (tree ...)**

Die maximale **Datensatz-ID** wurde überschritten. Es können keine weiteren Datensätze in der Tabelle eingefügt werden.

- **Record unpack error: field too long (file ...)**

Ungültige Feldlänge. Beim Entpacken eines Datensatzes wurde ein zu langes Alphafeld festgestellt.

- **Record unpack error: invalid block number (file ...)**

Unzulässige Fragmentnummer. Beim Entpacken eines Datensatzes wurde eine unzulässige Fragmentnummer ermittelt.

- **Record unpack error: invalid data length (file ...)**

Unzulässige Datenlänge. Beim Entpacken eines Datensatzes wurde eine unzulässige Datenlänge festgestellt.

- **Record unpack error: invalid key length (file ...)**

Ungültige Schlüssellänge. Beim Entpacken eines Datensatzes wurde eine ungültige Schlüssellänge festgestellt.

- **Record unpack error: invalid record ID (file ...)**

Ungültige Datensatz-ID. Beim Entpacken eines Datensatzes wurde eine ungültige Datensatz-ID festgestellt.

- **Record unpack error: invalid subrecord number (file ...)**

Ungültige Teildatensatznummer. Beim Entpacken eines Datensatzes wurde ein nicht vorhandener Teildatensatz festgestellt.

## Kontakt

- **Record unpack error: too many fields (file ...)**

**Ungültige Anzahl der Felder.** Beim Entpacken eines Datensatzes wurde eine ungültige Anzahl von Feldern festgestellt.

- **Text unpack error**

**Beim Entpacken eines Textes ist ein Fehler aufgetreten.** Der Text kann in der Regel noch gelöscht, wenn auch nicht mehr gelesen werden.

## Server Anfragefehler

Bei Anfragefehlern handelt es sich um irreguläre Zustände, die durch eine fehlerhafte Kommunikation ausgelöst wurden. In der Folge bricht der Server die Kommunikation mit dem entsprechenden Client ab.

- **Automatic Diagnosis & Recover failed (code ...)**

Automatische Diagnose mit Recover fehlgeschlagen. Möglicherweise steht nicht genügend RAM für eine Diagnose mit Recover zur Verfügung.

- **Automatic Diagnosis & Recover finished**

Automatische Diagnose mit Recover beendet. Das Diagnoseresultat befindet sich in der Datei <Datenbankname>.dgn.

- **Automatic Diagnosis & Recover started** Automatische

Diagnose mit Recover gestartet.

- **CRC error**

Ungültige Prüfsumme. Bei einem empfangenen Datenpaket stimmt die CRC-Prüfsumme nicht. Das Datenpaket wurde vom Kommunikationsmedium verfälscht. Dies kann durch Inkompabilitäten, Software- oder Hardware-Fehlern ausgelöst werden.

- **Data block length overflow (...)**

Ungültige Datenlänge. Der Client hat einen Datenblock mit unzulässiger Datenlänge gesendet.

- **Data structure damaged (code ...)**

Datenstruktur beschädigt. Beim Laden der Datenstruktur wurde ein Fehler festgestellt.

- **Free table: rebuild finished**

Die Reorganisation der Tabelle der freien Segmente wurde beendet.

- **Free table: rebuild started**

Die Reorganisation der Tabelle der freien Segmente wurde gestartet.

- **Invalid asynchronous command received (code ...)**

Ungültige Anfrage. Während der Schlüsselreorganisation hat der Client ein unzulässiges Anfragekommando gesendet.

## Kontakt

- **Invalid command received (code ...)**

**Ungültiges Kommando.** Ein Client hat einen unbekannten Befehl an den Server geschickt.

- **Invalid data key length received (...)**

**Ungültige Schlüssellänge.** Der Client hat einen Datenblock mit unzulässiger Schlüssellänge gesendet.

- **Invalid file number received (...)**

**Ungültige Dateinummer.** Die Client-Anfrage enthält eine nicht vorhandene Dateinummer.

- **Invalid filter data received**

**Ungültiger Filter.** Der Client hat unzulässige Daten für einen Zugriffsfilter gesendet.

- **Invalid tree number received (...)**

**Ungültige Baumnummer.** Die Client-Anfrage enthält eine ungültige Baumnummer.

- **Invalid tree root ...**

**Baumwurzel ungültig.** Beim Zugriff auf einen Baum der Datenbank wurde eine ungültige Baumwurzel festgestellt. Es sollte eine Diagnose mit Recover durchgeführt werden.

- **Invalid tree segment ...**

**Ungültiges Datensegment.** In einer Baumstruktur wurde ein nicht zugehöriges Datensegment entdeckt.

- **Key not found (file ..., key ...)**

**Schlüssel nicht vorhanden.** Beim Ändern oder Löschen eines Datensatzes wurde ein zugeordneter Schlüssel nicht gefunden. Es sollte eine Schlüsselanalyse und -reparatur oder eine Schlüsselreorganisation durchgeführt werden.

- **Key without record ...**

**Schlüssel ohne Datensatz.** Beim Datenzugriff wurde ein Schlüssel gefunden, zu dem kein Datensatz existiert. Es sollte eine Schlüsselanalyse und -reparatur oder eine Schlüsselreorganisation durchgeführt werden.

- **Pre-selection not active**

**Vorauswahl ohne Selektionskriterien.** Vor der Durchführung einer Selektionsvorauswahl sind keine Selektionskriterien registriert worden.

- **Record fragment sequence error (file ...)**

**Falsches Datensatzfragment.** Beim Einfügen oder Ändern von Datensätzen hat der Client ein unzulässiges Datensatzfragment gesendet.

- **Segment data block too large ...**

**Datenblock zu groß.** Innerhalb der Verarbeitung wurde ein zu großer Datenblock gefunden.

## Kontakt

- Segment structure damaged ...  
**Segmentstruktur beschädigt.** Beim Zugriff auf ein Segment der Datenbank wurde eine beschädigte Segmentstruktur festgestellt. Es sollte eine Diagnose mit Recover durchgeführt werden.
- Too many filter items specified  
**Zu viele Filterkriterien.** Der Client hat versucht, mehr als 100 Filterkriterien zu registrieren (RecFilterAdd()).
- Tree structure repaired ...  
**Baumstruktur repariert.** Bei einer Diagnose mit Recover wurde eine defekte Baumstruktur wiederhergestellt.
- Unexpected record write error (file ... result ...)  
**Fehler beim Speichern.** Beim Ändern oder Einfügen eines Datensatzes konnte ein Datenfragment nicht gespeichert werden. Es sollte eine Schlüsselanalyse und -reparatur oder eine Schlüsselreorganisation durchgeführt werden.

## Server Ein-/Ausgabefehler

- Cache Information File is not valid  
Die Pufferbelegungsdatei wird nicht eingelesen, weil sie nicht zur Datenbank passt, sie größer ist als der eingestellte Datenbankpuffer, oder sie beschädigt ist. (Siehe CacheSave)
- Cache Information File is valid  
Die Pufferbelegungsdatei ist gültig. Es wird begonnen sie einzulesen. (Siehe CacheSave)
- Cache Information File saved (... segments)  
Pufferbelegung wurde erfolgreich gesichert. In Klammern wird die Anzahl der gesicherten Segmente angegeben. (Siehe CacheSave)
- Cache Information File write failed (...)  
Fehler beim Sichern der Pufferbelegung. Der Fehler wird in Klammern angegeben. (Siehe CacheSave)
- Cache loading aborted  
Das Einlesen der Pufferbelegung wurde abgebrochen. Dies tritt auf, wenn der Datenbankserver gestoppt wird, bevor der Puffer vollständig wiederhergestellt wurde. (Siehe CacheSave)
- Cache loading completed (... in ... / read rate .../s)  
Das Einlesen der Pufferbelegung wurde erfolgreich abgeschlossen. Wenn Daten vorhanden waren, wird in Klammern die Menge, die Dauer, sowie die Leserate angegeben. (Siehe CacheSave / CachePreload)
- Cache preload started  
Keine gültige Pufferbelegungsdatei vorhanden, die Datenbank wird in den Puffer eingelesen. (Siehe CachePreload)

## Kontakt

- **Database expansion failed**

Datenraum kann nicht erweitert werden. Die Datenbank kann nicht mehr vergrößert werden, da kein Plattspeicher mehr frei ist. Das Problem lässt sich lösen, indem, auf dem Datenträger Platz geschaffen wird.

- **Database free space exhausted**

In der Datenbank sind keine freien Segmente mehr vorhanden. Üblicherweise wird bereits vor dem Auftreten dieser Meldung der Datenraum erweitert. Zur Vermeidung von potentiellen Datenverlusten wird die Datenbankoperation und die laufende Transaktion abgebrochen sowie der Benutzer abgemeldet.

- **Database reopen in read-write mode failed (...)**

Die Datenbank konnte nicht im "Lesen-Schreiben"-Modus geöffnet werden. Möglicherweise liest noch ein anderer Prozess die Datenbankdatei oder es liegt ein Problem beim Dateisystem vor. (Siehe [Blog](#))

- **Database unavailable (...)**

Auf die Datenräume der Datenbank kann nicht mehr zugegriffen werden.

- **No area updates for more than ... hour(s)**

In die Datenbank sind seit mehr als den angegebenen Stunden keine Updates mehr geschrieben worden. Die Meldung tritt erstmalig nach 12 Stunden auf und wird stündlich wiederholt. Tritt dieses Problem auf, muss das weitere Vorgehen mit dem [vectorsoft-Support](#) abgesprochen werden. In der Regel wird zur Analyse die Log-Datei der Datenbank benötigt.

- **No update events for more than ... hour(s)**

Die Datenbank hat für mehr als die angegebenen Stunden keine [Update-Ereignisse](#) verarbeitet, die in die Transaktionslog-Dateien geschrieben wurden. Die Meldung tritt erstmalig nach einer Stunde auf und wird stündlich wiederholt. Tritt dieses Problem auf, muss das weitere Vorgehen mit dem [vectorsoft-Support](#) abgesprochen werden. In der Regel wird zur Analyse die Log-Datei der Datenbank benötigt.

- **Transaction file read fault**

Beim Lesen der temporären Transaktionsdatei ist ein Fehler aufgetreten.

- **Transaction file write fault**

Schreibfehler in Transaktionsdatei. Es ist ein Fehler beim Schreiben in die temporäre Transaktionsdatei aufgetreten. Möglicherweise liegt ein Problem des Dateisystems vor, oder die Rechte des Servers sind nicht ausreichend. Dieser Fehler kann auch durch eine volle Festplatte verursacht worden sein.

- **Transaction log open failed (...)**

Die Transaktions-Logdatei konnte nicht geöffnet werden. Ein neues Transaktions-Log (\*.TL?) konnte nicht angelegt werden. Der Fehlerwert des Betriebssystems wird in Klammern angegeben. Möglicherweise liegt ein Problem des Dateisystems vor, oder die Rechte des Servers sind nicht ausreichend.

## Kontakt

- **Transaction log write fault**

Die temporäre Transaktionsdatei konnte nicht geschrieben werden. Beim Schreiben in die temporäre Transaktionsdatei ist ein Fehler aufgetreten. Der Fehlerwert des Betriebssystems wird in Klammern angegeben. Möglicherweise liegt ein Problem des Dateisystems vor, oder die Rechte des Servers sind nicht ausreichend. Es muss in jedem Fall ausreichen Plattenplatz vorhanden sein.

### Server Hot-Standby

- **Communication fault (...)**

Während der Kommunikation ist ein Fehler aufgetreten. Der Fehler wird in Klammern angegeben.

- **Connect from ...**

Es wurde eine Verbindung von der angegebenen IP-Adresse aufgebaut.

- **Connection test to ... failed**

Ist in der Datenbankeinstellung neben der IP-Adresse der Direktverbindung auch die IP-Adresse des LANs angegeben (siehe Installation der Hot-Standby Option), wird vom Master-System die Erreichbarkeit des Slave-Systems auch über die LAN-Adresse überprüft. Kann das Slave-System nicht erreicht werden, wird diese Warnung in die Protokolldatei geschrieben.

- **Connect to ...**

Es wurde eine Verbindung zum Managerprozess des Standby-Systems aufgenommen.

- **Connect to ... failed (...)**

Die Verbindung zum Managerprozess auf dem angegebenen Rechner ist nicht zustande gekommen. Der aufgetretene Fehler wird in Klammern angegeben. Der Fehler wird nur einmal in 24 Stunden protokolliert.

- **Database has lower update volume counter**

Beim Start des Hot-Standby-Betriebs wurde festgestellt, dass der Updatezähler kleiner ist als der der Standby-Datenbank.

- **Database has older timestamp**

Beim Start des Hot-Standby-Betriebs wurde festgestellt, dass der Zeitstempel älter ist als der der Standby-Datenbank.

- **Database HSB update failure (...)**

Die regelmäßige Übertragung von Updates an die Standby-Datenbank wurde unterbrochen. Der Fehler wird in Klammern angegeben.

- ◆ Insufficient free database cache - Der Sekundärserver puffert empfangene Datenbank-Updates im Datenbank-Cache. Von dort werden sie in die Datenbank geschrieben. Sind über 90 Prozent vom Cache mehr als eine Minute belegt, meldet das sekundäre System diesen Fehler und die Verbindung wird getrennt. Mögliche Ursachen für dieses Problem sind eine übermäßige Systemauslastung oder ein zu gering konfigurierter

## Kontakt

Datenbank-Cache auf dem Sekundärserver.

- Database HSB updates started

Die regelmäßige Übertragung von Updates an die Standby-Datenbank wurde gestartet.

- Database HSB updates stopped

Die regelmäßige Übertragung von Updates an die Standby-Datenbank wurde angehalten.

- Database is not synchronous

Beim Start des Hot-Standby-Betriebs wurde festgestellt, dass die beiden Datenbanken nicht synchron sind.

- Database is synchronous

Die Datenbanken auf dem Primär- und dem Sekundärsystem sind synchron.

- Database name is undefined

Der symbolische Name der Datenbank ist nicht angegeben. Der Hot-Standby-Betrieb kann nur bei Datenbanken mit einem symbolischen Namen eingerichtet werden.

- Database shutdown requested due to HSB master-master conflict Einer der beiden Datenbankprozesse hat festgestellt, dass beide Prozesse die Datenbank als Master geöffnet haben. Beide Datenbanken werden geschlossen und mit einer Login-Sperre versehen. Der Administrator muss anhand der Protokolldateien feststellen, welche Datenbank weiter verwendet werden soll. Die Direktverbindung der Server sollte überprüft werden.

- Database synchronisation locked

Die Synchronisations sperre der Datenbank wurde gesetzt.

- Database synchronisation required (Timestamps: master [...] slave [...] flags ...)

Eine Abweichung der Zeitstempel ist der Grund für eine Synchronisation der Datenbanken. Die Zeitstempel der Datenbanken werden in eckigen Klammern angegeben. Die flags dienen zur weiteren Analyse und können vom Support ausgewertet werden.

- Database user login locked due to synchronisation lock

Es wurde eine Login-Sperre gesetzt, weil keine Synchronisation aufgrund einer Synchronisations-Sperre stattfinden konnte.

- Database user login unlocked (HSB standby server not available)

Die Login-Sperre aufgrund einer nicht durchführbaren Synchronisation wurde manuell entfernt.

- Disconnect from ...

Die Verbindung wurde von dem Rechner mit der angegebenen IP-Adresse getrennt.

## Kontakt

- **Loopback connect refused**

Der Primär- und der Sekundär-Server wurden mit der gleichen Lizenz gestartet. Zum Starten des Sekundär-Servers wird keine Lizenz benötigt (vgl. [Installation der Hot-Standby Option](#)).

- **Message format invalid (...)**

Die übertragene Nachricht entspricht nicht dem korrekten Format. Möglicherweise sind bei einem Update des Programmstandes nicht alle Dateien ersetzt worden oder die Kommunikation funktioniert nicht fehlerfrei.

- **Message timeout**

Ein Message-timeout tritt auf, wenn das sekundäre System eine Minute keine Updates oder keep-alive-packets vom Primärsystem empfängt oder das Primärsystem drei Minuten keine Nachrichten an das sekundäre System versenden kann, beziehungsweise das sekundäre System keine Empfangsbestätigung zurücksendet. Die Verbindung wird in beiden Fällen abgebrochen. Mögliche Ursachen könnten übermäßig starke Systemauslastungen sein.

- **Missing license option**

Die Lizenz verfügt nicht über eine Hot-Standby-Option.

- **Release mismatch (... / ...)**

Die Release-Nummern der beiden CONZEPT 16-Server sind nicht identisch. Die Releases werden in Klammern angegeben. Auf beiden Systemen muss der gleiche Release installiert sein.

- **Server address not specified**

Die Adresse des anderen Servers wurde nicht eingetragen. Der Betrieb von Hot-Standby ist ohne diese Adresse nicht möglich.

- **Server mode mismatch**

Beide Server sind entweder als Master oder Slave konfiguriert (siehe [Hot-Standby - Einstellungen](#)).

- **Standby database in use**

Der Datenbankprozess auf dem anderen System wird gerade gestartet.

- **Standby database is readonly**

Die Datenbank des Slave-Systems ist in den Datenbank-Einstellungen auf "Read-Only" gesetzt.

- **Standby database process is running in active mode**

Der Datenbankprozess auf der Standby-Seite ist als Master aktiv. Der Eintrag erfolgt als Fehler, da sich an beide Datenbankprozesse Clients anmelden können.

- **Standby database process is still running**

Der Slave-Datenbankprozess ist noch aktiv. Der Master-Datenbankprozess versucht die Datenbank zu öffnen. Die vorhergehende Instanz des Slave-Prozesses hat die Datenbank aber noch geöffnet.

## Kontakt

- **Standby database process is terminating**  
Der Slave-Datenbankprozess beendet sich gerade.
- **Synchronisation aborted**  
Die Synchronisation der Datenbank wurde abgebrochen. Die Datenbank wird gelöscht.
- **Synchronisation completed (... in ... / transfer rate .../s)**  
Die Synchronisation der Datenbanken ist erfolgreich abgeschlossen. Die übertragenen Datenmenge, die Dauer der Synchronisation und die Übertragungsrate werden in Klammern angegeben.
- **Synchronisation: damaged segment**  
Die Datenbank weist einen Defekt auf Segmentebene auf. Auf der Datenbank sollte eine Diagnose mit Recover durchgeführt werden.
- **Synchronisation: database read fault**  
Beim Lesen der Datenbank auf dem Master-System ist ein Fehler aufgetreten. Das Problem liegt möglicherweise beim Dateisystem des Servers.
- **Synchronisation: database write fault**  
Beim Schreiben der Datenbank auf dem Standby-System ist ein Fehler aufgetreten. Dies kann zum Beispiel durch zu wenig freien Speicherplatz, mangelnden Rechte oder Problemen mit dem Dateisystem auf dem Standby-Server ausgelöst werden.
- **Synchronisation failure (...)**  
Bei der Synchronisation ist ein Fehler aufgetreten. Der in Klammern angegebene Fehler erlaubt nähere Rückschlüsse auf die Ursache. Die Datenbank, die nicht synchronisiert werden konnte, wird gelöscht.
- **Synchronisation: open failed (...)**  
Beim Öffnen der Datenbank zur Synchronisation ist ein Fehler aufgetreten. Der in Klammern angegebene Fehler erlaubt nähere Rückschlüsse auf die Ursache.
- **Synchronisation started**  
Die Synchronisation der Datenbank wird gestartet. Die Master-Datenbank befindet sich zu diesem Zeitpunkt im Backup-Modus.

## SOA-Service

- **Connection error (...)**  
Es kam zu einem Verbindungsfehler zwischen dem SOA-Service und dem Task-Prozess. Der Fehler wird in Klammern angegeben.
- **Procedure library load failed**  
Die zu ladende Bibliothek (siehe SOA-Service - Konfigurationsdatei) konnte nicht geladen werden.

## Kontakt

- Procedure runtime error [...]

Bei der Durchführung der Ereignisfunktion ist ein Laufzeitfehler aufgetreten.

- Socket TLS certificate load failed

Die Zertifikatsdatei (siehe SOA-Service - Konfigurationsdatei) wurde nicht gefunden oder sie enthält kein Zertifikat.

- Socket TLS private key load failed

Die Datei des privaten Schlüssels wurde nicht gefunden oder sie enthält keinen RSA-Schlüssel (siehe SOA-Service - Konfigurationsdatei).

- Socket TLS private key mismatch

Der private Schlüssel passt nicht zum Zertifikat.

- Start of task process [...] failed (...)

Der angegebene Task konnte nicht gestartet werden. Der aufgetretene Fehler wird in Klammern angegeben.

- Task process [...] launched

Der angegebene Task wurde gestartet.

- User overflow (...)

Die Benutzeranmeldung wurde wegen der Überschreitung des Benutzerlimits abgelehnt. Die Anzahl der zu diesem Zeitpunkt angemeldeten Benutzern wird in Klammern angegeben.

- VSO initialisation failure (...)

Fehler beim Laden der graphischen Benutzeroberbibliothek (`c16_objw.dll`) in der erweiterten Version (siehe SOA-Service - Konfigurationsdatei) des SOA-Service. In Klammern wird ein Fehlerhinweis ausgegeben.

## Laufwerkstreiber

- Driver not installed

Der Gerätetreiber ist nicht installiert. Die Installation kann mit Hilfe der Installationsroutine durchgeführt werden.

- Driver started successfully (...)

Der Laufwerkstreiber wurde erfolgreich gestartet. Die Versionsnummer des Gerätetreibers wird in Klammern angegeben.

- Drive stopped by user

Der Laufwerkstreiber wurde angehalten. Diese Meldung wird eingetragen, wenn im Resultat der DrvInit-Nachricht ErrDrvQuit zurückgegeben wird. Eine erneute Initialisierung wird erst bei einem Neustart des Laufwerkstreibers durchgeführt.

- Initialisation canceled by user

Die Initialisierung des Laufwerkstreibers wurde durch das Programm abgebrochen. Diese Meldung erfolgt, wenn das Resultat der DrvInit-Nachricht

## Kontakt

einen Fehler zurückgegeben hat. Die Initialisierung wird nach einer Minute erneut versucht.

- **Initialisation failed (...)**

Die Initialisierung des Laufwerks ist fehlgeschlagen. Wird in dem Ereignis **DrvInit** kein Laufwerksbuchstabe und kein Freigabename spezifiziert, wird in Klammern Required drive information not specified ausgegeben.

- **...: Socket communication fault (...)**

Bei dem angegebenen Ereignis (siehe **Ereignisse des Laufwerkstreibers**) ist es zu einem Kommunikationsfehler mit dem SOA-Task gekommen. Der Fehler wird in Klammern angegeben.

- **Unable to create temporary file (...)**

Die temporäre Datei konnte nicht angelegt werden. Der aufgetretene Fehler steht in Klammern.

- **Unable to create temporary folder (...)**

Das angegebene temporäre Verzeichnis konnte nicht angelegt werden. Der aufgetretene Fehler steht in Klammern.

## Datenbankreplikation

- **Replication file open failed (...)**

Die Replikationsdatei konnte nicht geöffnet werden. Der Fehlerwert des Betriebssystems wird in Klammern angegeben.

- **Replication file write fault**

Die Replikationsdatei konnte nicht geschrieben werden. Beim Schreiben in die Replikationsdatei ist ein Fehler aufgetreten. Möglicherweise liegt ein Problem des Dateisystems vor, oder die Rechte des Servers sind nicht ausreichend. Es muss in jedem Fall ausreichen Plattenplatz vorhanden sein.

## Dateizugriffs-Fehler:

Dateizugriffsfehler treten zusammen mit anderen Fehlermeldungen auf. Kann zum Beispiel eine Datenbank nicht geöffnet werden, wird dies mit einem Grund in Klammern protokolliert. Der Grund kann einer der folgenden sein:

- **Access denied**

Eine Datei konnte nicht geöffnet werden, weil der Zugriff verweigert wurde. Die Rechte des Benutzers müssen überprüft bzw. das Read-Only-Attribut der Datenbank-Datei entfernt werden.

- **Current directory cannot be removed** Das

Verzeichnis kann nicht entfernt werden.

- **File lock violation**

Die Datei ist von einem anderen Prozess gesperrt.

## Kontakt

- **File not found**

Die Datei wurde nicht gefunden.

- **File open failed**

Die Datei konnte nicht geöffnet werden.

- **File read fault**

Fehler beim Lesen der Datei.

- **File Set Pointer error**

Fehler beim Positionieren in der Datei.

- **File sharing violation**

Die Datei befindet sich bereits von einem anderen Prozess im Zugriff.

- **File write fault**

Fehler beim Schreiben der Datei.

- **Invalid drive**

Das angegebene Laufwerk ist nicht korrekt.

- **Invalid file format**

Das Dateiformat ist nicht gültig.

- **Invalid file handle**

Der verwendete Datei-Deskriptor ist nicht gültig.

- **Invalid file name**

Der Dateiname ist nicht gültig.

- **Invalid file size**

Die Datei hat eine ungültige Größe.

- **Path not found**

Der Pfad wurde nicht gefunden.

- **Too many open files**

Es sind bereits zu viele Dateien geöffnet.

### Socket-Fehler:

Socket-Fehler treten zusammen mit anderen Fehlermeldungen auf. Bricht zum Beispiel die Verbindung zum Hot-Standby-System ab, wird dies mit einem Grund in Klammern protokolliert. Der Grund kann einer der folgenden sein:

- **Connect to proxy failed**

Es konnte keine Verbindung mit dem Proxy aufgenommen werden.

## Kontakt

- **Host could not be resolved**

Der Hostname konnte nicht aufgelöst werden.

- **Hostname unknown**

Der Hostname oder die IP-Adresse ist unbekannt.

- **Invalid Proxy reply received**

Die Antwort des Proxy ist ungültig. Möglicherweise wird die Authentifizierungsmethode nicht unterstützt.

- **Proxy read failed**

Bei der Kommunikation mit dem Proxy ist ein Lesefehler aufgetreten.

- **Proxy unknown**

Der Proxy-Server wurde nicht gefunden.

- **Proxy user authorization failed**

Die Authentifizierung beim Proxy ist fehlgeschlagen, weil der Benutzer oder das Passwort ungültig ist.

- **Proxy write failed**

Bei der Kommunikation mit dem Proxy ist ein Schreibfehler aufgetreten.

- **Socket address unknown**

Die Adresse ist ungültig.

- **Socket bind failed**

Das Anbinden des Sockets an den Port ist fehlgeschlagen. Die Ursache ist meistens, dass der Port bereits belegt ist. (Siehe [Blog](#))

- **Socket close failed**

Der Socket konnte nicht geschlossen werden.

- **Socket connection failed**

Es konnte keine Verbindung zum Zielhost aufgebaut werden.

- **Socket connection refused**

Die Verbindungsanfrage wurde durch den Proxy abgelehnt.

- **Socket creation failed**

Der Socket konnte nicht angelegt werden.

- **Socket is not connected**

Die Socket-Verbindung wurde vor einer anstehenden Übertragung wieder geschlossen.

- **Socket listen failed**

Der Socket konnte nicht in den Listen-Zustand versetzt werden.

- **Socket read failed**

Beim Lesen des Sockets ist ein Fehler aufgetreten.

## Kontakt

- **Socket read overflow**

Beim Lesen des Sockets wurden zu viele Daten zurückgegeben.

- **Socket read underflow**

Beim Lesen des Sockets wurden zu wenig Daten zurückgegeben.

- **Socket select failed**

Bei der Abfrage der Socket-Verbindung ist ein Fehler aufgetreten.

- **Socket thread creation failed**

Ein neuer Thread für die Socket-Verbindung konnte nicht initialisiert werden. Eventuell konnte der Datenbankserver die notwendigen Systemressourcen nicht anfordern.

- **Socket write failed**

Beim Schreiben des Sockets ist ein Fehler aufgetreten.

- **TCP/IP initialisation failed**

Der TCP/IP-Protokollstack konnte nicht initialisiert werden.

### Kommunikationsfehler mit einem Lizenzserver

Bei einigen Protokolleinträgen für die Lizenz mit Softwareschutz (internetbasierten Lizenz) werden weitere Informationen zu einem Fehler in Klammern angegeben. Folgende Zusatzinformationen können angegeben werden:

- **Data overflow**

Es wurden zu viele Daten gesendet. Möglicherweise kam es zu einem Übertragungsfehler.

- **Hash value mismatch**

Der übertragene Hash-Wert ist ungültig. Möglicherweise kam es zu einem Übertragungsfehler.

- **Invalid header ID**

Der Header der Datei ist ungültig. Möglicherweise wurde keine Lizenz-Identitätsdatei verwendet.

- **Invalid license key**

Es wurde ein unbekannter Lizenzschlüssel übermittelt.

- **Invalid license number**

Es wurde eine falsche Lizenznummer übermittelt.

- **Invalid license production time**

Es wurde der falsche Produktionszeitpunkt für die Lizenz übermittelt.

- **Invalid license status**

Die Lizenz besitzt nicht den korrekten Status.

## Kontakt

- **Invalid reply**

Die Antwort des Lizenzservers gehört nicht zur letzten Abfrage.

- **Invalid request type**

Die Antwort vom Lizenzserver ist nicht korrekt.

- **Multiple License Identity files found**

Es wurden mehrere Lizenz-Identitätsdateien (\*.idn) gefunden. Es darf nur eine Datei vorhanden sein.

- **No license key**

Es wurde kein Lizenzschlüssel für die Lizenznummer gefunden.

- **No valid response**

Die HTTP-Rückgabe ist nicht 200 (Ok) oder die Länge der Nachricht bzw. der Content-Type ist nicht korrekt.

- **Out of memory**

Der notwendige Hauptspeicher konnte nicht angefordert werden.  
Möglicherweise steht nicht genug Hauptspeicher zur Verfügung.

- **Service not available**

Der Dienst zum Aktualisieren der Lizenzdatei konnte nicht erreicht werden oder hat einen Fehler zurückgegeben. Die Serversoftware versucht eine erneute Anfrage nach einer Stunde.

- **Time deviation exceeds 24 hours**

Die Systemzeit des lokalen Systems weicht über 24 Stunden von der Systemzeit des Lizenzservers ab.

- **Unknown encryption type**

Die Informationen sind in einem unbekannte Typ kodiert.

- **Unknown hash type**

Es wurde ein unbekannter Hash-Typ zur Kodierung verwendet.

- **Unknown license number**

Es wurde eine unbekannte Lizenznummer übermittelt.

## Unbestimmte Fehlerwerte

- **unknown error code ...**

Der vom Betriebssystem angegebene Fehlerwert ist unbekannt. Der unbekannte Fehlerwert wird ebenfalls übermittelt.

## Serverinterne Fehler

## Kontakt



**Fehler dieser Kategorie bezeichnen eine Inkonsistenz innerhalb der Verarbeitung von Anfragen durch den Server. Solche Fehler deuten oft auf Instabilitäten des Serverrechners hin. Bitte teilen Sie solche Fehler auch dem vectorsoft-Support mit.**

- generic ...
- SegAllocTrcFile (Entry)
- SegCheckConflict (S)
- SegCheckConflict (User)
- SegCheckConflict (User)
- SegCheckConflict (User NULL ...)
- SegCheckConflict (X)
- SegCheckDQ (WaitQ)
- SegEOT (SOP-wait conflict) - user id ...
- SegFreeMgr::DataOut (table corrupted)
- SegFreeMgr::prvAllocate (map allocate failed ...)
- SegFreeMgr::prvAllocate (map not found ...)
- SegFreeMgr::prvRelease (map release failed ...)
- SegFreeTrcFile (Entry)
- SegKill (S-lock)
- SegMoveEntry
- SegNew (segment in use ...)
- SegRBT (Level)
- SegRBT (Lock)
- SegRBT (User)
- SegReadCache (Seg0) - segment ...
- SegRWArea (Entry) - slot ...
- SegScanLRU (Empty)
- SegSetUserLockInfo (Overflow)
- SegUnLinkHash
- SegUnLock (underflow)
- SegUpdate (AFI list)
- SegUpdate (AFI list clear)
- SegUpdate (mark list ...)
- SegUpdate (SLV list ...)
- SegUpdate (UPD list)
- SegWriteCache (Seg0)
- TrcRc (Clear)
- TrcRc (UsrID)

## Kontakt

### CONZEPT 16-Server - Diagnoseresultate

Beschreibung der Resultate nach der Durchführung einer Datenbankdiagnose Umfang der Datenbankdiagnose

Der CONZEPT 16-Datenbankserver verfügt über Funktionen zur Überprüfung und Reparatur von Datenbanken (Siehe [Datenbankdiagnose](#)). Das Ergebnis einer solchen Funktion wird, wenn möglich durch den Client angezeigt und in jedem Fall in der Datei <Datenbankname>\*.dgn im Verzeichnis der Datenbank abgelegt. Das Diagnoseresultat kann mit einem Texteditor betrachtet werden. Am Anfang des Resultates steht der Name der diagnostizierten Datenbank, die Art der Diagnose und Datum sowie Uhrzeit des Diagnosestarts.

Im Diagnoseresultat werden alle überprüften Baumstrukturen der Datenbank aufgeführt. In der Datenbank sind dabei die Bäume mit den Nummern 0-39 und 65001-65013 fest vergeben. Bäume mit den Nummern 40 - 65000 werden dynamisch an Dateien und Schlüssel vergeben.

Folgende festvergebene Bäume sind definiert:

- 0 Baumverzeichnis
- 1 Textinhalte
- 2 Prozedurtexte
- 3 Prozedurcode
- 4 Hilfetexte
- 5 Notizbücher
- 6 Notizbücher
- 7 Notizbücher
- 8 Notizbuchnamen
- 10 Druckertreiber
- 11 Druckertreiber
- 12 Bemerkungen
- 13 Prozedurinformationen
- 14 Listenformate
- 15 Selektionen
- 16 Selektionen
- 17 Menüs
- 18 Zählerstände
- 19 Selektionen
- 20 Transfers
- 21 Transfers
- 22 Transfers
- 23 Textgruppen
- 24 Textverzeichnis
- 25 Benutzer
- 26 Benutzer

## Kontakt

27      Benutzer  
28      Dateiparameter  
29      Basisparameter  
30      Datenstruktur  
31      Datenstruktur  
32      Datenstruktur  
33      Datenstruktur  
36      Funktionstastenbelegungen  
37      Hilfetexte  
38      Datenstruktur  
**65001** Listenelemente  
**65002** Listenelemente  
**65003** Listenelemente  
**65004** Zugriffslisten  
**65005** Druckertreiber  
**65006** Zugriffslisten  
**65007** Zugriffslisten  
**65008** Prozedurinformationen  
**65010** GUI-Objekte  
**65011** GUI-Objekte  
**65012** Daten binärer Objekte  
**65013** Daten binärer Objekte  
Diagnoseresultate

Im Diagnoseresultat stehen hinter dem Namen des diagnostizierten Bereichs in eckigen Klammern die ermittelte Anzahl von Einträgen und das Diagnoseergebnis für diesen Bereich.

Baumstruktur        42 Schlüssel        1 / 2 [ 2235 ] : ok

Dieser Eintrag bedeutet, dass der Baum 42 zum zweiten Schlüssel der Datei 1 gehört, 2235 Einträge hat und die Struktur des Baumes in Ordnung ist. Bei Fehlern steht der Fehlerwert in Klammern hinter dem Ergebnis.

Baumstruktur        43 Schlüssel        1 / 2 [ 1006 ] : Defekt (5) Baumstruktur        44

Am Ende des Diagnoseresultates wird der diagnostizierte Status der Datenbank und die Dauer der Diagnose vermerkt. Folgende Diagnosergebnisse sind möglich:

- Ok

Es wurden keine Fehler oder Defekte in der Datenbank festgestellt.

- Defekt

Die Datenbank beinhaltet Fehler.

- Korrigiert

## Kontakt

**Das Recover hat Fehler korrigiert.**

### Verhalten bei Diagnoseoptionen

Wird eine Datenbankdiagnose mit der Recover-Option durchgeführt, werden Defekte in der Datenbank repariert. Bereiche, in denen der Recover-Lauf erfolgreich Reparaturen vorgenommen hat, werden im Diagnosresultat als "Repariert" oder "Korrigiert" markiert. Konnte der Defekt nicht behoben werden, steht im Resultat weierhin "Defekt". Dabei wird zwischen den folgenden Defekten unterschieden:

- Fehler in der Speicherverwaltung

Fehler in diesem Bereich werden ohne Auswirkungen auf den Datenbestand korrigiert.

- Baumdefekte

Bei einem defekten Baum wird dieser neu aufgebaut. Dabei werden alle Daten, die in der Datenbank zu diesem Baum gefunden werden, verwendet. Je nach Defekt des Baumes und Zustand der Datenbank kann dabei ein Datenverlust auftreten.

- Fehler bei Schlüsselwerten

Die Zählerstände von Schlüsselwerten können durch das Recover korrigiert werden, Schlüssel-Kollisionen (die doppelte Vergabe eines Schlüssels) können dagegen nicht automatisch beseitigt werden. In diesem Fall muss der betroffene Datenbereich exportiert, in der Datenbank gelöscht und wieder importiert werden.

- Fehler bei binären Objekten

Defekte oder logische Zuordnungsfehler bei binären Objekten werden vom Recover durch das Löschen der defekten Objekte beseitigt.

Werden bei der Diagnoseoption "Schlüsselanalyse und -reparatur" Fehler festgestellt oder Korrekturen vorgenommen, tauchen diese folgendermaßen im Diagnoseresultat auf. das Diagnoseresultat lautet bei behobenen Fehlern "Korrigiert".

- unvollständige Sätze

Bei Datensätzen in der angegebenen Datei ist der Satzanfang vorhanden, es fehlen aber Inhalte der Teildatensätze oder der Felder.

- gelöschte Sätze

Diese Sätze konnten nicht korrekt entpackt werden und wurden gelöscht.

- gelöschte Satzfragmente

Hierbei wurden Bruchstücke von Datensätzen entfernt, die nicht mehr zugeordnet werden konnten.

- eingefügte Schlüssel

Dies ist die Anzahl von fehlenden und eingefügten Schlüsselwerten.

- gelöschte Schlüssel

## Kontakt

Diese Schlüssel waren abweichend vom Datensatz oder konnten keinem Datensatz zugeordnet werden.

- Wertekollisionen

Die Anzahl von doppelten Werten bei eindeutigen Schlüsseln. Diese Fehler kann die Schlüsselanalyse nicht selbstständig korrigieren. Ein Administrator mit Zugriff auf die Datenbankstruktur muss daher diese Fehler beheben.

### Behandlung von Defekten in der Datenbank

Bei der Diagnose können Baumdefekte unterschiedlicher Fehlerklassen ermittelt werden. Bei den Fehlernummern 1 bis 4, 21 bis 24 und 36 bis 37 handelt es sich um Fehler im Aufbau des Baumes. Die Fehlernummern 5 bis 9 und 32 bis 35 kennzeichnen Fehler in einem einzelnen Segment des Baumes. Da die Diagnose des Baumes beim ersten ermittelten Fehler abgebrochen wird, können durchaus mehrere unterschiedliche Fehler in einem Baum enthalten sein.

Grundsätzlich können Baumdefekte zu Datenverlust führen. Sofern nicht auf eine Datensicherung zurückgegriffen wird, muss der Baumdefekt zunächst mittels eines Recover-Laufs behoben werden. Die Datenbank darf keinesfalls mit defekten Baumstrukturen weiterbetrieben werden, da dies unter Umständen zu Folgedefekten führen kann. Je nach Art des betroffenen Baumes sind verschiedene Maßnahmen sinnvoll:

- Baumverzeichnis

Bei Defekten am Baum 0 sollte auf eine Datensicherung zurückgegriffen werden, da unter Umständen Teile des Datenbestandes nicht mehr erreichbar sind.

- Bäume, die Definitionen beinhalten

Bei Datenverlust in den Definitionen sollte die aktuelle Datenbankdefinition eingespielt werden. Liegt keine aktuelle Definition vor, kann diese mittels OEM-Kit aus einer Sicherungsdatenbank erstellt werden. Sofern die Bäume 1 und 24 betroffen sind (Texte) kann auch ein Verlust von Anwendungsdaten auftreten.

- Bäume, die Datensätze beinhalten

Bei einem Datenverlust fehlen anschließend Datensätze in der betroffenen Datei. Nach dem Recover ist immer eine Reorganisation aller Schlüssel der Datei notwendig. Durch Vergleich der Datensatzanzahl in der Datei vor und nach Recover und Reorganisation lässt sich feststellen, ob tatsächlich Datensätze verlorengegangen sind.

- Bäume, die einen Index enthalten

Dieser Fall ist unkritisch. Nach dem Recover muss lediglich der betroffene Schlüssel reorganisiert werden. Ein Defekt eines Schlüsselbaumes hat keine Auswirkungen auf den Datenbestand.

- Speicherverwaltung

## **Kontakt**

**Defekte in der Speicherverwaltung (Fehlernummer 11 bis 19)** können durch ein Recover behoben werden. Es ist aber auf jeden Fall sicherzustellen, dass auf dem Datenträger der Datenbank noch ausreichend freier Speicher verfügbar ist.

- **Binäre Objekte**

**Fehler bei binären Objekten (Fehlernummern 50 bis 52)** können durch ein Recover behoben werden, wobei allerdings Datenverluste entstehen können.

- **Baumzuordnung**

**Fehler in der Baumzuordnung** können nicht durch ein Recover beseitigt werden. Hier muss entweder eine Datensicherung verwendet werden oder eine manuelle Reparatur der Datenbank durch den Datenbankhersteller vectorsoft erfolgen.

- **Satz- und Schlüsselanzahl der Dateien**

Bei abweichenden Werten wird der jeweilige Schlüssel mit "abweichend" markiert. Durch eine Reorganisation des betreffenden Schlüssels kann das Problem beseitigt werden.

## Kontakt

### Datenbank-Replikation

Beschreibung des Vorgangs zum replizieren einer Datenbank Siehe  
[Replikations](#)

[Utility](#), [Blog](#)

Die Datenbank-Replikation bietet eine Möglichkeit die in einer Datenbank angefallenen Veränderungen als inkrementelle Backups zu sichern. Diese inkrementellen Backups lassen sich in eine Kopie der Datenbank wieder einspielen.

### Vorgehen zum Einstellen der Replikation

In der Datenbank-Konfiguration des CONZEPT 16-Servers (siehe auch [Konfiguration des Servers](#)) müssen die Einträge Replikationspfad und Replikationsintervall eingestellt werden. Dabei ist zu beachten, dass der Replikationspfad möglichst auf einem anderen Laufwerk des Servers liegen sollte, da ein großes Datenvolumen entstehen kann.

Anschließend muss die Datenbank ein Mal geöffnet und wieder geschlossen werden und eine Kopie der Datenbank angefertigt werden. Die Kopie wird ebenfalls in dem Replikationspfad abgelegt.



Die Kopie der Datenbank darf niemals direkt geöffnet werden, sonst ist ein Einspielen der Replikationssätze nicht mehr möglich. Um die Kopie der Datenbank zu öffnen ist es daher notwendig eine weitere Kopie der Kopie anzufertigen.

Fortan werden Replikationsdatensätze erzeugt, die mit dem [Replikations Utility](#) in die Kopie der Datenbank eingespielt werden können.

### Beschreibung der Replikationssätze

Der Name eines Replikationssatzes ist folgendermaßen aufgebaut:

<Datenbankname>\_<Datenbankseriennummer>\_<Replikationsseriennummer>.tlg

Datenbankseriennummer

Fortlaufende Seriennummer der Datenbank

Die fortlaufende Seriennummer der Datenbank wird immer dann erhöht, wenn die Datenbank in einen kopierbaren Status versetzt wird (Datenbank geschlossen oder Backup-Zustand).

Replikationsseriennummer Fortlaufende Seriennummer des Replikationssatzes

Die fortlaufende Seriennummer der Replikationssätze wird jedes Mal erhöht, wenn ein Replikationssatz erzeugt wird. Wenn die Seriennummer der Datenbank erhöht wird, wird die Replikationsseriennummer auf 1 zurückgesetzt.

Weiterhin enthält ein Replikationssatz einen eigenen Zeitstempel, Informationen über den zuvor ausgespielten Replikationssatz (Seriennummern und Zeitstempel) und den Zeitstempel der Datenbank. Aufgrund dieser Informationen wird sichergestellt, dass die Replikationssätze vollständig vorhanden sind und nicht bereits importierte Replikationssätze erneut in die Datenbank eingespielt werden.

### Hinweise

## Kontakt

Nach folgenden Aktionen ist es Notwendig eine neue Kopie der Originaldatenbank anzufertigen, um weiterhin Replikationssätze einspielen zu können:

- Datenbankoptimierung
- Datenbankdiagnose mit Recover
- Rollback der Datenbank

## Kontakt

### CONZEPT 16 - Replikations Utility

#### Beschreibung der Funktionen des Replikations Utilities

##### Replikation,

Siehe Replikation

(Blog)

Das Replikations Utility ist ein Kommandozeilenprogramm, mit dem man die erzeugten Replikationssätze einer Datenbank in die Kopie der Datenbank einspielen kann.

**c16\_serv\_rpu\_w32.exe** Windows Betriebssysteme (32-Bit)

**c16\_serv\_rpu\_w64.exe** Windows Betriebssysteme (64-Bit)

**c16\_serv\_rpu\_l32** Linux Betriebssysteme (32-Bit)

**c16\_serv\_rpu\_l64** Linux Betriebssysteme (64-Bit)

Dem Programm müssen zur Durchführung der Kommandos bestimmte Argumente an der Komandozeile übergeben werden. Die Kommandos sind für alle Betriebssysteme identisch. Die Argumente hängen vom jeweiligen Kommando ab.

```
c16_serv_rpu_xxx <Kommando> [<Argumente>] [-delete] [-compress]
```

 Bei der Angabe von Argumenten wird unter Linux zwischen Groß- und Kleinschreibung unterschieden.

Der Rückgabewert gibt Aufschluss darüber, ob das Kommando erfolgreich durchgeführt wurde. Wird das Replikations Utility in einer Script-Datei verwendet, kann dieser Rückgabewert über den Errorlevel des Betriebssystems ermittelt und ausgewertet werden. Zusätzlich zum Rückgabewert wird auf der Komandozeile auch eine entsprechende Information ausgegeben.

 Das Replikations Utility kann auf jedem beliebigen Rechner ausgeführt werden. Eine CONZEPT 16-Lizenz wird dazu nicht benötigt. Jedoch ist zu empfehlen das Programm auf einem 64-Bit-Betriebssystemen mit möglichst viel Arbeitsspeicher auszuführen, um die Daten schnell einzuspielen.

Folgende Kommandos können an das Replikations Utility übergeben werden:

- info

Informationen einer Datenbank, eines Replikationssatzes oder einer Transaktionslog-Datei ermitteln.

- verify

Einen Replikationssatz oder eine Gruppe von Replikationssätzen auf Vollständigkeit und Einspielbarkeit in die Datenbank überprüfen.

- replay

Einen Replikationssatz oder eine Gruppe von Replikationssätzen in eine Datenbank einspielen.

- salvage

Einen (unvollständigen) Replikationssatz in eine Datenbank einspielen.

- pack

Einen Replikationssatz oder eine Gruppe von Replikationssätzen

## Kontakt

**zusammenpacken.**

## Kontakt

### Fehlermeldungen und Logeinträge vom Replikations Utility

Fehlermeldungen und Logeinträge bei der Ausführung des Replikations Utility Die folgenden Fehlermeldungen und Logeinträge können bei der Verwendung des Replikations Utility generiert werden.

### Mögliche Fehler

- **No input file specified**

Es wurde keine Quelldatei (Datenbank oder TLG-Datei) angegeben.

- **No database file specified**

Es wurde keine Datenbank angegeben.

- **Database replication flag not set**

Das Replikationsflag ist in der Datenbank nicht gesetzt. Dieses muss in der Konfiguration des Servers gesetzt werden. Anschließend muss die Datenbank geöffnet werden.

- **Database incomplete (SYNC)**

Die Quelldatenbank ist nicht geschlossen.

- **Database not closed correctly**

Die Quelldatenbank wurde nicht korrekt geschlossen.

- **Replication file (...) open failed (...)**

Die Replikationsdatei (...) konnte aus dem angegebenen Grund nicht geöffnet werden.

- **Replication file (...) damaged: invalid block ID**

Die Replikationsdatei (...) ist beschädigt. Sie enthält eine ungültige Block-ID.

- **Replication file (...) damaged: block sequence error**

Die Replikationsdatei (...) ist beschädigt. Die Reihenfolge der Blöcke ist abweichend.

- **Replication file (...) damaged: invalid data size**

Die Replikationsdatei (...) ist beschädigt. Die Größe des Datenblockes ist ungültig.

- **Replication file (...) damaged: invalid data ID**

Die Replikationsdatei (...) ist beschädigt. Die ID des Datenblockes ist ungültig.

- **Replication file (...) damaged: invalid segment size**

Die Replikationsdatei (...) ist beschädigt. Die Größe des Segments ist ungültig.

- **Replication file (...) damaged: CRC error**

Die Replikationsdatei (...) ist beschädigt. Die Checksumme der Replikationsdatei ist nicht korrekt.

- **Replication file (...) is incomplete**

Die Replikationsdatei (...) ist unvollständig. Sie kann nur mit dem Kommando

## Kontakt

salvage eingespielt werden.

- Database fingerprint don't match

Die Replikationsdatei passt nicht zu der angegebenen Datenbank.

- Replication file (...) serial number don't match

Der Zähler oder der Zeitstempel der Replikationsdatei (...) passt nicht zu dem vorhergehenden Replikationssatz oder der Datenbank.

- Replication file (...) has duplicate data

Durch die gewünschte Aktion würden die Daten doppelt vorhanden sein.

Weitere Fehler können im Bereich der Dateibearbeitungsfehler liegen.

### Einträge im Datenbank-Log

- Replication started on serial number (...)

Einspielen von Replikationssätzen wurde gestartet. Die Datenbank hatte vorher den Zählerstand (...).

- Replication completed with serial number (...)

Einspielen von Replikationssätzen wurde erfolgreich beendet. Die Datenbank hat den Zählerstand (...).

- Replication finished incomplete with serial number (...)

Einspielen eines unvollständigen Replikationssatzes wurde erfolgreich beendet. Die Datenbank hat den Zählerstand (...).

### CONZEPT 16 - Replikations Utility

info

Replikations

Utility,

Fehler und

**info - Informationen einer Datenbank, eines Replikationssatzes oder einer Transaktionslog-Datei ermitteln**

Syntax:

```
c16_serv_rpu_w64.exe info <Name der ca1-Datei> | <Name der tlg-Datei> | <Name der tl?-Datei>
```

**Ausgabe der Informationen einer Datenbank, eines Replikationssatzes oder einer Transaktionslog-Datei.** Bei einer Datenbank erhält man Informationen darüber, wann die Datenbank initialisiert wurde, und welche Seriennummer die Datenbank hat. Weiterhin kann man sehen welcher Replikationssatz zuletzt aus- oder eingespielt wurde. Bei Replikationsdateien erhält man Informationen zu den Datenbank-Zeitstempeln aus der Replikationssatz entstanden ist, sowie die eigene Replikationsseriennummer und die Seriennummer des vorhergehenden Replikationssatzes. Transaktionslog-Dateien enthalten Informationen zu den Datenbank-Zeitstempeln und der eigenen Seriennummer.

Rückgabe:

- 0 - Kommando erfolgreich ausgeführt.
- 1 - Fehler aufgetreten.

Beispiele:

Informationen einer Datenbank ermitteln.

```
c16_serv_rpu_w64.exe info MyDatabase.ca1
```

Informationen eines Replikationssatzes ermitteln.

```
c16_serv_rpu_w64.exe info MyDatabase_000010_000001.tlg
```

Informationen einer Transaktionslog-Datei ermitteln.

```
c16_serv_rpu_w64.exe info MyDatabase.tl2
```

## Kontakt

### CONZEPT 16 - Replikations Utility

#### pack

##### Replikations Utility,

Siehe

##### Fehler und

**pack - Einen Replikationssatz oder eine Gruppe von Replikationssätzen zusammenpacken  
(konsolidieren)**

**Syntax:**

```
c16_serv_rpu_w64.exe pack <Name der tlg-Datei(en)> [-delete] [-compress]
```

**Mit diesem Befehl wird ein Replikationssatz oder eine Gruppe von Replikationssätzen zusammengepackt. Dazu ist es notwendig, dass die Replikationsdateien vollständig sind. Eine Gruppe von Replikationsdateien kann durch Wildcards (?, \*) angegeben werden. Vor dem Packen der Replikationssätze wird geprüft, ob diese eingespielt vollständig sind und zusammen gehören. Einzelne Replikationssätze können nur mit der Option compress gepackt werden. Mit der Option delete werden die Quell-Replikationssätze nach erfolgreicher Durchführung gelöscht.**



**Unter Linux ist bei der Verwendung der Wildcards zu beachten, dass der Suchausdruck in Anführungszeichen zu setzen ist, da andernfalls die Shell den Stern interpretiert.**



**Ein mit pack gepackter Replikationssatz kann nur mit anderen Replikationsdateien erneut gepackt werden.**

**Rückgabe:**

- 0 - Kommando erfolgreich ausgeführt.
- 1 - Fehler aufgetreten.

**Beispiele:**

**Einen Replikationssatz komprimieren.**

```
c16_serv_rpu_w64.exe pack MyDatabase_000010_000001.tlg -compress
```

**Alle Replikationssätze mit den Datenbankseriennummern 10 - 19 zusammenpacken und die einzelnen Replikationssätze anschließend löschen.**

```
// Windows c16_serv_rpu_w64.exe pack MyDatabase_00001?_* .tlg -delete // Linux ./c16_serv_rpu_164 pac
```

## Kontakt

### CONZEPT 16 - Replikations Utility

replay

Replikations  
Utility,

Siehe

Fehler und

**replay** - Einen Replikationssatz oder eine Gruppe von Replikationssätzen in eine Datenbank einspielen

Syntax:

```
c16_serv_rpu_w64.exe replay <Name der cal-Datei> <Name der tlg-Datei(en)> [-delete]
```

Mit diesem Befehl wird ein Replikationssatz oder eine Gruppe von Replikationssätzen in eine Datenbank eingespielt. Dazu ist es notwendig, dass die Replikationsdateien vollständig sind und die Zeitstempel, sowie die Seriennummern des ersten Replikationssatzes zu den gespeicherten Seriennummern in der Datenbank passen. Eine Gruppe von Replikationsdateien kann durch Wildcards (?, \*) angegeben werden. Vor dem Einspielen der Replikationssätze wird geprüft, ob diese eingespielt werden können. Mit der Option delete werden die Quell-Replikationssätze nach erfolgreicher Durchführung gelöscht.



Unter Linux ist bei der Verwendung der Wildcards zu beachten, dass der Suchausdruck in Anführungszeichen zu setzen ist, da andernfalls die Shell den Stern interpretiert.

Rückgabe:

- 0 - Kommando erfolgreich ausgeführt.
- 1 - Fehler aufgetreten.

Beispiele:

Einen Replikationssatz in die Datenbank einspielen.

```
c16_serv_rpu_w64.exe replay MyDatabase.cal MyDatabase_000010_000001.tlg
```

Alle Replikationssätze mit den Datenbankseriennummern 10 - 19 in die Datenbank einspielen und die eingespielten Replikationssätze anschließend löschen.

```
// Windows c16_serv_rpu_w64.exe replay MyDatabase.cal MyDatabase_00001?_*.*.tlg -delete // Linux ./c16
```



Um Replikationssätze in eine Datenbank einzuspielen, darf diese Datenbank nicht seit dem letzten Kopiervorgang geöffnet werden, da sonst die Seriennummer der Datenbank verändert wird.

### CONZEPT 16 - Replikations Utility

salvage

Replikations  
Utility,

Siehe

Fehler und

**salvage - Einen (unvollständigen) Replikationssatz in eine Datenbank einspielen**

Syntax:

```
c16_serv_rpu_w64.exe salvage <Name der cal-Datei> <Name der tlg-Datei> [-delete]
```

Mit diesem Befehl wird ein Replikationssatz, der unvollständig ist in eine Datenbank eingespielt. Ist die Replikationsdatei vollständig, kann er auch mit diesem Kommando eingespielt werden. Vor dem Einspielen des Replikationssatzes wird geprüft, ob dieser eingespielt werden kann. Unvollständige Replikationsdateien haben die Endung \$++. Diese muss vor dem Einspielen auf tlg geändert werden. Hat der Datenbankserver die Datei noch im Zugriff, so kann diese nicht umbenannt werden. Ist die Option delete angegeben, wird der Quell-Replikationssatz nach erfolgreicher Durchführung gelöscht. Ein unvollständiger Replikationssatz kann entstehen, wenn die Datenbank nicht geschlossen wird, bevor der Server terminiert wird.



Nach dem Einspielen eines unvollständigen Replikationssatzes ist kein weiteres Einspielen möglich.

Rückgabe:

- 0 - Kommando erfolgreich ausgeführt.
- 1 - Fehler aufgetreten.

Beispiele:

Einen unvollständigen Replikationssatz einspielen und anschließend löschen.

```
c16_serv_rpu_w64.exe salvage MyDatabase.cal MyDatabase_000010_000001.++$ -delete
```

## Kontakt

### CONZEPT 16 - Replikations Utility

verify

Replikations

Utility,

Siehe

Fehler und

**verify** - Einen Replikationssatz oder eine Gruppe von Replikationssätzen auf Vollständigkeit und Einspielbarkeit in die Datenbank überprüfen

Syntax:

```
c16_serv_rpu_w64.exe verify [<Name der cal-Datei>] <Name der tlg-Datei(en)>
```

Mit diesem Befehl wird ein Replikationssatz oder eine Gruppe von Replikationssätzen auf Vollständigkeit überprüft. Eine Gruppe von Replikationsdateien kann durch Wildcards (?, \*) angegeben werden. Wird vor dem Replikationssatz / den Replikationssätzen eine Datenbank angegeben, wird zusätzlich überprüft, ob die Dateien in die Datenbank eingespielt werden können.



Unter Linux ist bei der Verwendung der Wildcards zu beachten, dass der Suchausdruck in Anführungszeichen zu setzen ist, da andernfalls die Shell den Stern interpretiert.

Rückgabe:

- 0 - Kommando erfolgreich ausgeführt.
- 1 - Fehler aufgetreten.

Beispiele:

Einen Replikationssatz überprüfen.

```
c16_serv_rpu_w64.exe verify MyDatabase_000010_000001.tlg
```

Alle Replikationssätze mit den Datenbankseriennummern 10 - 19 überprüfen.

```
// Windows c16_serv_rpu_w64.exe verify MyDatabase_00001?_* .tlg // Linux ./c16_serv_rpu_164 verify "M
```

Prüfen, ob ein Replikationssatz in die Datenbank eingespielt werden kann.

```
c16_serv_rpu_w64.exe verify MyDatabase.cal MyDatabase_000010_000001.tlg
```

Prüfen, ob alle Replikationssätze mit den Datenbankseriennummern 10 - 19 in die Datenbank eingespielt werden können.

```
// Windows c16_serv_rpu_w64.exe verify MyDatabase.cal MyDatabase_00001?_* .tlg // Linux ./c16_serv_rp
```

## Kontakt

### CONZEPT 16-Server - Hot-Standby

#### Beschreibung der CONZEPT 16-Server Hot-Standby Option Praxis 1

(Blog),

Praxis 2

Siehe (Blog),

Virtuelle

Server

(Blog)

Die CONZEPT 16-Server Hot-Standby Option ist eine Hochverfügbarkeitslösung. Es handelt sich dabei um ein Sicherheitskonzept, welches aus zwei miteinander verbundenen Serversystemen besteht. Der CONZEPT 16-Server mit Hot-Standby-Option verwendet einen zweiten Server als automatische Reserve für den Fall, dass das Primärsystem ausfällt. Die Serversysteme können hierbei sowohl in der verwendeten Systemarchitektur als auch im verwendeten Betriebssystem unterschiedlich konfiguriert sein.

Auf dem Primärsystem liegt die originale Datenbank, welche mit den Clients geöffnet und bearbeitet werden kann. Beim ersten Öffnen der Datenbank auf dem Primärsystem wird diese komplett auf das Sekundärsystem übertragen. Im laufenden Datenbankbetrieb werden dann alle abgeschlossenen Transaktionen vom Primärsystem in die Datenbank auf dem Sekundärsystem transferiert. Die Übertragung wird über eine direkte Verbindung beider Rechner mit Gigabit-Netzwerkkarten oder vergleichbarem durchgeführt.

Auf dem Sekundärsystem ist also zu jeder Zeit eine aktuelle und konsistente Datenbank vorhanden. Fällt nun der Primärserver aus, oder wird der Primärserver aufgrund von Wartungsarbeiten gestoppt, übernimmt automatisch der Sekundärserver die Aufgaben des Primärservers. Ein Arbeiten an der CONZEPT

16-Datenbank ist daher auch im Falle eines Primärserverausfalls uneingeschränkt möglich.

Die Beschreibung zur Hot-Standby Option gliedert sich in die folgenden Kapitel:

- Installation der Hot-Standby Option
- Arbeitsweise des Hot-Standby
- Einrichten von Sperren
- Servicefunktionen im Hot-Standby Betrieb
- Checkliste Hot-Standby



Beim Betrieb des Servers auf einer virtuellen Maschine sind gerade im Zusammenhang mit der Hot-Standby-Option einige Besonderheiten zu beachten. Hinweise dazu befinden sich im Abschnitt Hinweise zum Betrieb auf virtuellen Maschinen.

## Kontakt

### CONZEPT 16-Server - Installation der Hot-Standby Option

#### Beschreibung zur Installation der Hot-Standby Option des CONZEPT 16-Servers

##### Serversysteme und Netzanbindung

Für die CONZEPT 16 Hot-Standby Option werden zwei CONZEPT 16-Datenbank-Server benötigt. Diese werden über eine direkte Gigabit-Ethernet Verbindung oder vergleichbarem miteinander verbunden. Zusätzlich müssen diese Rechner ebenfalls vom Netzwerk aus erreicht werden können. In jedem Rechner müssen also mindestens zwei Netzwerkanschlüsse installiert sein, einen zur Kommunikation mit dem Netzwerk und einen zur Kommunikation mit dem anderen Serversystem. Weitere Hardware wird für den Hot-Standby Betrieb nicht benötigt.

Arbeitet eines der Systeme zusätzlich als Fileserver, ist zu beachten, dass beim Ausfall des Systems die Dateien des CONZEPT 16-Clients nicht mehr zur Verfügung stehen. Das kann vermieden werden, wenn der CONZEPT 16-Client auf den lokalen Festplatten der Clientsysteme abgelegt wird. Ein Versionsabgleich könnte dann zum Beispiel durch eine Verteilungssoftware oder Skript bei der Anmeldung vorgenommen werden (siehe auch FAQ).

Die Systeme können sich von der Leistungsfähigkeit und sogar vom verwendeten Betriebssystem unterscheiden. Im Normalbetrieb wird das Sekundärsystem zwar wesentlich geringer belastet als das Primärsystem, allerdings muss es bei einem Ausfall des Primärsystems in der Lage sein, die anfallenden Benutzeranfragen und Transaktionen ausreichend schnell bearbeiten zu können. Auf dem Sekundärsystem können neben dem CONZEPT 16-Server natürlich auch weitere Dienste (z. B. Druckdienste oder Job-Server) betrieben werden.

Die Hot-Standby-Option funktioniert nur, wenn das Sekundärsystem die gleichen Voraussetzungen für den Einsatz der verwendeten Lizenz erfüllt, wie das Primärsystem. Die Voraussetzungen beziehen sich auf die Anzahl der logischen Prozessoren, das Betriebssystem und ob es sich um ein 64-Bit System handelt. Hot-Standby kann zum Beispiel nicht eingesetzt werden, wenn das Primärsystem unter einem Windows-Betriebssystem läuft und das Sekundärsystem unter Linux, wenn eine Standard-Edition verwendet wird (Diese läuft nur unter Windows oder Linux). Beim Einsatz einer Advanced-Edition gibt es keine Probleme, da diese Lizenz unter Windows und Linux betrieben werden kann.

### Datenbankserver

Auf beiden Rechnern muss der CONZEPT 16-Server in der gleichen Version inklusive Lizenzdatei installiert werden. Bei einer Lizenz mit Hardwareschutz muss der Dongle-Treiber mindestens auf dem Primärsystem installiert sein. Auf dem sekundären Server wird der Dongle-Treiber nur benötigt, um bei einem längeren Ausfall des Primärsystems gegebenenfalls ein Umstecken des Dongles zu ermöglichen. Bei einer Lizenz mit Softwareschutz darf sich die Lizenzidentitätsdatei (\*.idn) nur auf dem Primärsystem befinden. Ohne ein Umstecken des Dongles bzw. Kopieren der Lizenzidentitätsdatei können alle Hot-Standby Datenbanken auf dem Sekundärsystem noch weitere 60 Tage geöffnet werden. Die Zeitspanne wird ab dem letzten Kontakt, durch Aktualisierung der Datenbank, mit dem Primärsystem gemessen und ist unabhängig davon, ob der Server in der Zwischenzeit neu gestartet wurde. Danach ist der Login für neue Clients gesperrt.

## Kontakt

 Während dieser Zeit dürfen die Einstellungen der Datenbank (Name, Datenbank und die Adresse des Primärsystems (Serveradresse)) nicht verändert werden.

Zum Starten des Systems muss der Dongle (bei Verwendung einer Lizenz mit Hardwareschutz) beim Primärsystem aufgesteckt werden. Das Sekundärsystem wird ohne eine Lizenz gestartet.

 Wird der CONZEPT 16-Server mit Clients der Versionen 4.7 oder 5.0 betrieben, können Datenbanken nur solange geöffnet werden, wie der CONZEPT 16-Server nicht neu gestartet wurde und weniger als 60 Tage seit der letzten Aktualisierung durch das Primärsystem vergangen sind. Bei einem Neustart des Systems muss der Dongle auf das Sekundärsystem gesteckt oder die Lizenzidentitätsdatei auf das Sekundärsystem kopiert werden, damit sich Clients der Versionen 4.7 oder 5.0 mit einer Datenbank verbinden können.

Auf dem Sekundärserver kann eine zusätzliche Serverlizenz eingerichtet werden. So können auch Datenbanken geöffnet werden, die nicht für den Hot-Standby Betrieb eingerichtet wurden. Falls die 60-Tage Frist einer Hot-Standby Datenbank nach einem Primärserverausfall verstrichen ist, wird dann auch der Login nicht gesperrt, sondern die Lizenz des Sekundärservers verwendet. Dabei ist darauf zu achten, dass es nach Ablauf der primären Lizenz zu Einschränkungen in der Anzahl der Benutzer beziehungsweise des Funktionsumfangs kommen kann, wenn unterschiedliche Serverlizenzen verwendet werden. Die Lizenz des Sekundärservers benötigt dazu keine Hot-Standby-Option.

Besitzt der Sekundärserver keine eigene Lizenz, sollte in der Konfiguration des Servers die Option Hot-Standby-Betrieb ohne Dongle aktiviert werden.

 Um aufgetretene Probleme rechtzeitig zu erfahren, sollte unbedingt die automatische Mailbenachrichtigung (siehe Konfiguration des Servers) beim Primär- und beim Sekundärserver eingerichtet werden. Da die Benutzer nach einer kurzen Unterbrechung ihre Arbeit mit dem System fortsetzen können, ist in der Regel nicht damit zu rechnen, dass der Ausfall an den Systemadministrator gemeldet wird. Das Benachrichtigungssystem des CONZEPT 16-Servers informiert dann in jedem Fall umgehend von dem Wechsel zum Sekundärsystem.

### Einstellungen der Datenbanken

Die Einstellungen der Datenbanken erfolgt über die Web-Administration des Servers. Einstellungen können auch über ältere Clients (ab Version 5.1) vorgenommen werden. Es stehen dann aber nicht alle Einstellungen zur Verfügung. Ältere Versionen bekommen von dem Server keine Lizenz und können nicht gestartet werden. Im laufenden Betrieb wird eine Lizenz für die älteren Versionen vom Primärserver auf den Sekundärserver übertragen. Die Lizenz befindet sich im Hauptspeicher des Sekundärservers.

Die Datenbank muss auf beiden Serversystemen eingetragen werden. Der symbolische Name der Datenbank muss auf beiden Systemen gleich sein. Alle anderen Eintragungen können abweichen. Die physikalische Datenbankdatei darf nur auf dem Primärsystem abgelegt werden. Die Datenbank wird beim ersten Öffnen automatisch auf den Sekundärserver übertragen.

## Kontakt

**i** Im laufenden Hot-Standby-Betrieb empfängt der Sekundärserver Datenbankupdates vom Primärserver und puffert diese im Datenbankcache. Von dort werden sie in die Datenbank übertragen. Ist das sekundäre System zu stark ausgelastet und kann neue Transaktionen nicht aus dem Cache in die Datenbank schreiben, kann es zu einer Trennung der Verbindung kommen. Diese wird automatisch durchgeführt, wenn der Cache des sekundären Systems mehr als eine Minute zu über 90 Prozent belegt ist (Siehe auch [Database HSB Update Failure](#) in der Log-Datei des Datenbankservers). Da ein zu klein konfigurierter Datenbankcache auf dem sekundären System praktisch sofort zu diesem Zustand führen kann, ist unbedingt darauf zu achten, dessen Größe korrekt einzustellen.

Die Einstellungen für den Betrieb mit der Hot-Standby Option werden über die Administration des Servers vorgenommen. Die Einstellungen werden nach der Auswahl der Datenbank über den Link [Hot-Standby](#) aufgerufen. Optional kann auch die Datenraumtabellen des Datenbankservers direkt editiert werden.

Die folgenden Einstellungen können für den Hot-Standby-Betrieb vorgenommen werden:

The screenshot shows the CONZEPT 16 software interface for managing databases. The title bar reads "CONZEPT 16 - Datenbank & mehr". The address bar shows the URL "127.0.0.1:4745/area/hotstandby?ca1=CodeLibrary". The main navigation menu includes "CONZEPT 16-Server", "Status", "Datenbank" (selected), "Konfiguration", "Manager", "Einstellungen", "Backup", "Hot-Standby" (selected), "Service", and "Statistik". A sub-menu titled "Hot-StandBy Einstellungen für 'CodeLibrary'" is open, displaying various configuration options. These include:

Einstellung	Wert
Serveradresse	192.168.0.2;10.1.3.91
Sekundärserver	<input type="checkbox"/>
Automatische Aktivierung	<input type="checkbox"/>
Automatische Synchronisation	<input checked="" type="checkbox"/>
Timeout Verbindungsaufbau [Sekunden]	3
Verbindungs-Intervall [Minuten]	5
Automatische Abschaltung bei Master-Master Betrieb	<input checked="" type="checkbox"/>
Login sperren wenn Standby-Server nicht mehr verfügbar	<input type="checkbox"/>

A green checkmark icon with the text "Übernehmen" is located at the bottom right of the configuration area.

- Serveradresse

## Kontakt

Hier wird die IP-Adresse des anderen Servers angegeben. Erfolgt die Eintragung beim Primärserver, wird hier die IP-Adresse des Sekundärservers angegeben. In der Eintragung des Sekundärservers wird die IP-Adresse des Primärservers eingetragen. Bei diesen IP-Adressen handelt es sich um die Adressen der Netzwerkkarten der Direktverbindung. Es sollten keine Namen anstelle der numerischen IP-Adressen verwendet werden, da sonst im Falle von Problemen mit der Namensauflösung keine Verbindung zwischen den Servern hergestellt werden kann.

Die IP-Adresse der "normalen" Netzwerkverbindung muss nach einem Semikolon ebenfalls angegeben werden. Ist diese Adresse nicht vorhanden, können keine Statusinformationen über die normale Netzverbindung ausgetauscht werden. Bei einem Ausfall der Direktverbindung kann es dann vorkommen, dass ein gleichzeitiges Anmelden an Primär- und Sekundär-Server möglich ist. Durch die Angabe der "normalen" Netzwerkverbindung können Statusinformationen zwischen den Servern ausgetauscht werden und ein gleichzeitiges Anmelden an beide Datenbanken ist ausgeschlossen. Zudem wird die Erreichbarkeit des Sekundärsystems für die Clients durch das Primärsystem überprüft. Ist das Sekundärsystem nicht für die Clients erreichbar, zum Beispiel wegen Einstellungen der Firewall, wird eine entsprechende Warnung in die Protokolldatei geschrieben.



**Die IP-Adressen der Direktverbindung und die IP-Adressen der "normalen" Netzwerkverbindung müssen in unterschiedlichen Netzen liegen.**

- **Sekundärsystem**

Mit dieser Einstellung wird die Unterscheidung der beiden Server in "Primärsystem" und "Sekundärsystem" vorgenommen. Die Einstellung muss daher auf beiden Servern unterschiedlich sein. Die Einstellung muss nach der Installation nicht mehr verändert werden, da die Rolle des jeweiligen Servers ("Aktiv" oder "Standby") davon unabhängig ist.

- **Automatische Aktivierung**

Ist diese Option nicht gesetzt, wird durch eine Sperre verhindert, dass sich Benutzer direkt an die Datenbank anmelden können, wenn der aktive Server ausgefallen ist (siehe Einrichten von Sperren).

- **Automatische Synchronisation**

Ist diese Option nicht gesetzt, wird durch eine Sperre verhindert, dass die Datenbank nach einem Serverausfall automatisch wieder mit der aktiven Datenbank synchronisiert wird (siehe Einrichten von Sperren). Scheitert eine Synchronisation an dieser Sperre, wird zusätzlich die Login-Sperre gesetzt und ein entsprechender Eintrag in die Protokolldatei der Datenbank vorgenommen. Die Sperre muss durch den Administrator über die Funktion Freigabe für Login entfernen werden.

- **Timeout Verbindungsauflaufbau**

Beim Öffnen der Datenbank versucht der aktive Server eine Verbindung zum Standby-Server aufzubauen. Gelingt dies nicht in der hier angegebenen Zeitspanne, wird die Datenbank auf dem Standby-Server nicht aktualisiert. Der

## Kontakt

**nächste Verbindungsauftbau wird nach der Zeit, die in dem Feld "Verbindungs-Intervall" angegeben ist, initiiert. Der Timeout kann zwischen 3 Sekunden und 300 Sekunden (5 Minuten) betragen.**

**Ist der Standby-Server nicht mehr erreichbar, verzögert sich die Anmeldung des ersten Benutzers um den hier eingetragenen Wert. Die Ausführung von Serviceoperationen verzögert sich ebenfalls um diesen Wert.**

**Empfohlener Wert: 3 Sekunden.**

- **Verbindungs-Intervall**

**Kommt es beim Öffnen einer Datenbank nicht zu einer Verbindung zum Standby-Server, wird in regelmäßigen Abständen versucht diese Verbindung erneut herzustellen. Es können hier Wartezeiten zwischen 1 Minute und 15 Minuten eingestellt werden.**

**Falls die Verbindung zum Standby-Server abbricht, werden erneute Verbindungsversuche auch in diesen Zeitabständen durchgeführt. Empfohlener Wert: 5 Minuten.**

- **Automatische Abschaltung bei Master-Master Betrieb**

**Bei eingerichtetem Hot-Standby-Betrieb dürfen sich nie beide Datenbanken im gleichzeitig Master-Modus befinden, da sich in diesem Zustand Clients auf beiden Datenbanken arbeiten können und dadurch später Datenverluste entstehen.**

**Erkennt einer der beiden beteiligten Datenbankserver einen solchen Zustand und ist diese Option gesetzt, werden beide Datenbank geschlossen, die Benutzer ausgeloggt und die Datenbanken mit einer Login- und einer Synchronisations-Sperre versehen. In den Protokollen der Datenbanken wird der Fehler Database shutdown requested due to HSB master-master conflict eingetragen.**

**Um wieder mit der Datenbank arbeiten zu können, muss der Administrator anhand der Protokolle entscheiden, welches System sich fälschlicherweise im Master-Modus befand. Diese Datenbank muss umbenannt oder verschoben werden. Bei der richtigen Datenbank muss danach eine "Freigabe für Login" erfolgen.**

**Ein Master-Master-Betrieb kann beispielsweise auftreten, wenn die aktive Datenbank (Master) auf einer virtuellen Maschine läuft, die eingefroren wird. Übernimmt in diesem Fall das Standby-System die Masterrolle, sind nach dem Aufwecken des virtuellen Systems beide Datenbanken im Master-Modus.**

- **Login sperren wenn Standby-Server nicht mehr verfügbar**

**Wenn diese Option eingeschaltet ist, wird bei aktiven Hot-Standby-Betrieb eine bedingte Sperre in der Master-Datenbank eingetragen.**

**Diese Sperre verhindert das Öffnen der Datenbank im Master-Betrieb, wenn das Standby-System nicht erreichbar ist. Die Sperre kann über die Weboberfläche**

## Kontakt

des Servers angezeigt und auch entfernt werden.

Im normalen Betrieb hat die Sperre keine Auswirkungen, sie wird auch nicht auf die Standby-Datenbank übertragen. Die Sperre ist nur in folgendem Szenario wirksam:

Nach erfolgreichem Hot-Standby-Betrieb wird die Master-Datenbank geschlossen. Ist beim erneuten Öffnen der Datenbank der Standby-Server nicht verfügbar, kann sich kein Benutzer an der Datenbank anmelden. Sobald der Standby-Server wieder funktioniert, können sich wieder Benutzer an der Master-Datenbank anmelden. Fallen nacheinander zunächst der Primär- und anschließend der Sekundärserver aus, können sich erst dann wieder Benutzer an der Datenbank anmelden, wenn beide Server wieder gestartet sind und so die tatsächliche Master-Datenbank ermittelt werden kann.

Bei längerer Nichtverfügbarkeit des Standby-Systems muss die Sperre über die Weboberfläche des Masters entfernt werden. Die Sperre wird automatisch unwirksam, wenn die HSB-Adresse bei der Masterdatenbank entfernt wird und somit der Hot-Standby-Betrieb deaktiviert wird.



Für die Verbindungen des Clients zum Server kann eine Maximalzeit eingestellt werden, die der Client beim Verbindungsaufbau zum Server warten darf. Danach wird automatisch der Sekundärserver angesprochen. Weiteres dazu unter [Konfigurationsdatei / TcpConnectTimeout](#).

### *Beispiel zur Konfiguration der Hot-Standby Systeme*

Die Netzwerkkarten der Direktverbindung zwischen den beiden Serversystemen verwenden in dieser Konfiguration die Adressen 192.168.0.1 und 192.168.0.2. Die Adressen zur Kommunikation mit dem Netzwerk über die beiden Zweitnetzwerkkarten könnten dann beispielsweise 10.0.0.1 und 10.0.0.2 lauten.

#### Konfiguration des Primärservers

Datenbankname	Beispieldatenbank
Speicherort	c:\c16\ca1\Examples-1
Adresse des anderen Servers	192.168.0.2;10.0.0.2
Sekundärsystem	deaktiviert
Automatische Aktivierung	deaktiviert
Automatische Synchronisation	aktiviert
Timeout Verbindungsaufbau	3 Sekunden
Verbindungs-Intervall	5 Minuten
Automatische Abschaltung bei Master-Master Betrieb	aktiviert
Login sperren wenn Standby-Server nicht mehr verfügbar	deaktiviert

#### Konfiguration des Sekundärservers

Datenbankname	Beispieldatenbank
Speicherort	c:\c16\hotstandby\Examples-2
Adresse des anderen Servers	192.168.0.1;10.0.0.1

## Kontakt

<b>Sekundärsystem</b>	aktiviert
<b>Automatische Aktivierung</b>	aktiviert
<b>Automatische Synchronisation</b>	aktiviert
<b>Timeout Verbindungsaufbau</b>	3 Sekunden
<b>Verbindungs-Intervall</b>	5 Minuten
<b>Automatische Abschaltung bei Master-Master Betrieb</b>	aktiviert
<b>Login sperren wenn Standby-Server nicht mehr verfügbar</b>	deaktiviert
<b>Aufruf der Datenbank über eine Verknüpfung</b>	

Wenn eine Datenbank über eine Verknüpfung geöffnet werden soll, kann generell die Executable des Clients, gefolgt von den Parametern Datenbankserver, Datenbankaliasname, Datenbankbenutzer und Passwort, in der Verknüpfung angegeben werden. So kann der Anwender direkt in die für ihn vorgesehene Umgebung einsteigen. Beim Hot-Standby Betrieb würde dies aber bedeuten, dass der Client nur versuchen würde den Primärserver zu erreichen. Daher können als Datenbankserver zwei verschiedene Systeme, getrennt von einem + angegeben werden. Zwischen Primärserver, der Konjunktion und dem Sekundärserver dürfen dabei keine Leerzeichen vorhanden sein. Der Client versucht sich dann zunächst beim ersten angegebenen Server anzumelden. Ist dieser nicht erreichbar oder ist die Datenbank älter als die des anderen Servers, verbindet sich der Client automatisch mit dem zweiten Server.

### Beispiel:

```
c:\c16\client\c16_winc.exe dbserver+dbbackup codelibrary
```

oder

```
c:\c16\client\c16_winc.exe 10.0.0.1+10.0.0.2 codelibrary
```

## Kontakt

### CONZEPT 16-Server - Arbeitsweise des Hot-Standby

#### Beschreibung des Zusammenspiels zwischen Primär- und Sekundärserver bei CONZEPT 16-Hot-Standby Systemen

Nach der Installation der Datenbankserver und dem Eintragen der Datenbanken, ist die Datenbank nur auf dem primären Server vorhanden. Beim Öffnen der Datenbank auf dem Primärserver, wird durch diesen eine Verbindung zum Sekundärserver (Eintrag "Serveradresse") hergestellt. Ist der Sekundärserver nicht erreichbar, dauert das Öffnen der Datenbank bis das "Timeout Verbindungsauftbau" abgelaufen ist. Der Benutzer kann sich anschließend an der Datenbank anmelden, unabhängig davon, ob ein Sekundärserver gefunden wurde oder nicht.



Alle Ereignisse in Verbindung mit der Hot-Standby-Option (Aufbau einer Verbindung, Abbruch der Verbindung, Ausfall von Primär- oder Sekundärserver usw.) werden in der Log-Datei der Datenbank (<Datenbankname>.lgb) protokolliert. Diese Log-Datei wird im Verzeichnis der Datenbank abgelegt (siehe auch Architektur des Servers).

Der Primärserver fragt nach der Benutzeranmeldung und gelungenem Verbindungsauftbau zum Sekundärserver den Status der sekundären Datenbank ab. Da die Datenbank zu diesem Zeitpunkt noch nicht vorhanden ist (Status ist "unbekannt"), wird eine Synchronisation gestartet. Bei der Synchronisation wird die Datenbank in den Backup-Zustand (siehe Sicherungsereignisse) versetzt und anschließend werden alle Segmente der Datenbank an den Sekundärserver geschickt. Bricht während der Übertragung der Segmente die Verbindung ab (es konnten also nicht alle Segmente der Datenbank übertragen werden), wird die Datenbank wieder gelöscht. Eine Datenbank, die nicht alle Segmente enthält ist unbrauchbar.

Die neue Sekundärdatenbank wird mit einem Zeitstempel versehen, der um 100 Nanosekunden kleiner ist, als der Zeitstempel der Master-Datenbank auf dem primären Server. Der Zeitstempel wird aus der Systemzeit generiert und hat eine Auflösung von ca. 1 Millisekunde. Durch den Zeitstempel der Master-Datenbank und dem nicht natürlich entstehenden Zeitstempel der Standby-Datenbank kann später ermittelt werden, ob die Daten synchron sind. Die Datenbank liegt jetzt auf beiden Systemen vor.



Die Datenbanken auf dem Primärserver und dem Sekundärserver müssen sich immer durch den Zeitstempel unterscheiden. Damit nicht zwei identische Zeitstempel in den Datenbanken vorkommen, dürfen die Datenbanken nicht von einem Server auf den anderen kopiert werden. Das Verschieben einer Datenbank auf den anderen Server ist aber möglich. Der Zeitstempel der Datenbank kann über die Web-Administration angezeigt werden.

Solange die Datenbank vom Primärserver zum Sekundärserver übertragen wird, hat die Datenbank auf dem Sekundärserver den Status "Synchronisation". Die in der Zwischenzeit angefallenen Daten werden nach der Synchronisation in die primäre Datenbank übertragen. Die Datenbank wird also nach der Synchronisation wieder zum Schreiben geöffnet. Der Status der Primärdatenbank ist dann "offen", da sich ja gerade ein Benutzer angemeldet hat. Der Zustand der Sekundärdatenbank wird nun auf "Standby" gesetzt. An der Datenbank mit dem Status "Standby" können sich generell keine Benutzer anmelden.

## Kontakt

Ab diesem Zeitpunkt werden in regelmäßigen Abständen die neuen und geänderten Segmente der Primärdatenbank an das Sekundärsystem geschickt. Dabei erfolgt auch immer ein Abgleich der Zeitstempel der Datenbanken. Sendet das primäre System keine Update-Nachrichten mehr, erkennt das sekundäre System nach einem Timeout den Ausfall des Primärservers. Da während Update-Events keine Nachrichten an das Sekundärsystem versendet werden, signalisiert das Primärsystem seine Funktionstüchtigkeit mindestens alle drei Sekunden durch Keep-Alive-Pakete. Fällt das primäre System aus und es erfolgt ein Wechsel auf das Sekundärsystem, wird die Datenbank auf dem Sekundärsystem mit einem neuen Zeitstempel versehen. Anhand des Zeitstempels wird nach dem Schließen und erneuten Öffnen der Datenbanken festgestellt, ob die Datenbank auf dem Primärserver einen älteren Stand hat als die auf dem Sekundärsystem. Es ist daher wichtig, dass die Uhrzeiten des Primär- und des Sekundärservers weitestgehend synchron sind. Nach einem Wechsel auf das Sekundärsystem werden alle Funktionen insbesondere das weitere Arbeiten auf diesem System und der dort abgelegten Datenbank durchgeführt.



Weichen die Systemzeiten um mehr als eine Minute ab, wird die Uhrzeit des Standby-Systems an die Uhrzeit des aktiven Systems angepasst. Unterschiedliche Zeitzonen sind dabei irrelevant, da der Abgleich auf Basis der UTC-Zeit (Universal Time Coordinated) durchgeführt wird.

Sind bei den Einstellungen sowohl die Adresse der Direktverbindung, als auch die Adresse des "normalen" Netzwerkes angegeben, wird bei einem Ausfall der Direktverbindung der Status des anderen Servers über das normale Netzwerk abgefragt. So werden zum Beispiel Anmeldungen an das Sekundärsystem unterbunden, auch wenn keine Kommunikation über die Direktverbindung möglich ist. Zudem wird im laufenden Betrieb die Erreichbarkeit des Sekundärsystems für die Clients durch das Primärsystem überprüft. Kann das Sekundärsystem von einem Client nicht erreicht werden, werden entsprechende Warnungen in die Protokolldatei geschrieben.

Bei folgenden Operationen wird der Hot-Standby Betrieb unterbrochen und nach dem Ende der Operation eine erneute Synchronisation durchgeführt:

- Diagnose mit Recover oder Schlüsselanalyse
- Manuelle Datenraumerweiterung
- Manuelles Anlegen eines weiteren Datenraums
- Optimierung

## Kontakt

### CONZEPT 16-Server - Einrichten von Sperren

#### Beschreibung der Sperren im CONZEPT 16-Server Hot-Standby Betrieb

Das Verhalten beim Ausfall eines der beiden Hot-Standby Server wird auch von Sperren innerhalb der Datenbanken beeinflusst. Diese Sperren können durch das System automatisch gesetzt und im Client-Dialog "Service" manuell wieder entfernt werden (siehe auch Installation der Hot-Standby Option / Einstellungen).

- Login / Aktivierung

Diese Sperre wird beim Öffnen einer Datenbank im Standby-Modus gesetzt, falls die Option Automatische Aktivierung nicht ausgewählt ist. Die Sperre verhindert das Öffnen der Datenbank im aktiven Modus.

Falls der Standby-Server den Ausfall des aktiven Servers erkennt, schließt er die Standby-Datenbank. Versucht nun ein Benutzer die Datenbank zu öffnen, wird zunächst wieder nach dem aktiven Server gesucht. Kann dieser nicht gefunden werden, muss die Datenbank auf dem Standby-Server aktiviert werden. Ist die Aktivierungs-Sperre gesetzt, so muss diese zunächst entfernt werden, bevor sich Benutzer an die Datenbank anmelden können. Der Benutzer bekommt folgende Fehlermeldung:

Da in der Regel ein möglichst nahtloser Übergang vom Primärserver zum Sekundärserver gewünscht ist, sollte bei der Sekundärdatenbank die automatische Aktivierung aktiv sein. Sollen sich allerdings die Benutzer bei einem Ausfall des Primärservers beim Administrator melden, kann der Eintrag Automatische Aktivierung auch auf dem Sekundärsystem deaktiviert werden. Die Sperre in der Datenbank muss dann mit der Aktion Freigabe für Login wieder entfernt werden.



Mit der Aktion "Freigabe für Login" wird die Sperre für das Hot-Standby-System entfernt. Die Funktion "Login freigeben" wird verwendet, um die Aktion "Login sperren" aufzuheben. Diese Aktion wird verwendet, um keine neuen Anmeldungen an eine Datenbank zuzulassen, weil diese zum Beispiel für Wartungszwecke geschlossen werden soll.

In der Datenbank auf dem Primärserver sollte die automatische Aktivierung nicht ausgewählt sein. Für den Fall, dass der Sekundärserver ebenfalls abgeschaltet und der Primärserver vor dem Sekundärserver aktiviert wird, kann dadurch verhindert werden, dass die Benutzer mit einem alten Stand der Datenbank arbeiten.

- Synchronisation

Diese Sperre wird nur beim Öffnen der aktiven Datenbank gesetzt, wenn die Option Automatische Synchronisation nicht ausgewählt ist. Die Sperre verhindert das Überschreiben der jeweiligen Datenbank durch einen Synchronisations-Vorgang.

Falls der aktive Server anhand der Zeitstempel erkennt, dass die Standby-Datenbank auf einem veralteten Stand ist, wird diese mit der Datenbank auf dem aktiven Server synchronisiert. Falls die automatische

## Kontakt

Synchronisation deaktiviert wurde, muss zunächst die Sperre manuell entfernt werden (Servicefunktion Freigabe für Synchronisation).

Bei der Sekundärdatenbank muss die automatische Synchronisation aktiviert sein, da sonst kein korrekter Hot-Standby-Betrieb möglich ist. Auf der Primärdatenbank sollte die automatische Synchronisation verhindert werden. Der Sekundärserver würde nach einem erneuten Start des ausgefallenen Primärservers sofort wieder eine Verbindung herstellen und die Datenbanken abgleichen, ohne das zuvor eine vollständige Überprüfung des Primärsystems möglich wäre.

Ist die Ursache für den Primärserverausfall behoben, kann die Sperre über die Aktion Freigabe für Synchronisation wieder entfernt werden. Die Synchronisation der Datenbanken erfolgt dann beim nächsten Verbindungsauflauf durch den aktiven Server.

- **Rollback**

Diese Sperre wird beim Öffnen der aktiven als auch der Standby-Datenbank gesetzt. Die Sperre verhindert die automatische Durchführung eines Rollbacks beim Öffnen einer nicht korrekt geschlossenen Datenbank. Der Benutzer bekommt folgende Fehlermeldung angezeigt:

Beim Ausfall eines Servers (dies ist meist Grund für nicht korrekt geschlossene Datenbanken) übernimmt in der Regel das Standby-System die aktive Rolle, wodurch keine Notwendigkeit besteht, auf der ehemals aktiven Datenbank ein Rollback durchzuführen. Sollte aber zum Zeitpunkt des Ausfalls der Hot-Standby-Betrieb unterbrochen sein (beispielsweise wenn der Sekundärserver abgeschaltet ist), muss ein Rollback der Datenbank vorgenommen werden. Dazu muss vorher die Sperre entfernt werden (Servicefunktion Freigabe für Rollback).

Vor dem Entfernen einer Rollback-Sperre sollte in jedem Fall der Zustand und die Aktualität der Datenbanken auf Primär- und Sekundärsystem überprüft werden, um ein versehentliches Rollback auf einer nicht mehr aktuellen Datenbank auszuschließen. Die Zeitstempel der Datenbanken können über die Web-Administration ermittelt werden.

- **Standby-System nicht verfügbar**

Diese Sperre wird beim Öffnen der aktiven als auch der Standby-Datenbank gesetzt, wenn in den Einstellungen der Eintrag Login sperren wenn Standby-Server nicht mehr verfügbar gesetzt ist. Ist das zweite System (Primär- oder Sekundärserver) nicht erreichbar, kann die Datenbank nicht geöffnet werden. Der Benutzer erhält folgende Fehlermeldung:

Mit dieser Sperre kann verhindert werden, dass die Datenbank geöffnet wird, ohne den Zustand der zweiten Datenbank ermitteln zu können. Die Sperre kann mit der Servicefunktion Freigabe für Öffnen ohne Standby entfernt werden.

- **Datenbank im Standby-Modus**

## **Kontakt**

**Versucht ein Benutzer eine Datenbank zu öffnen, die im Slave-Modus betrieben wird, wird folgende Fehlermeldung angezeigt:**

**Der Benutzer muss die Datenbank auf dem Server öffnen, auf dem sie als "Master" betrieben wird.**

- **Datenbank ist durch den Administrator gesperrt**

**Datenbanken können durch den Administrator über die Servicefunktion Login sperren für weitere Anmeldungen gesperrt werden. Benutzer, die sich an die Datenbank anmelden wollen, bekommen folgende Fehlermeldung:**

**Bereits angemeldete Benutzer können abgemeldet werden, um zum Beispiel Wartungsfunktionen an der Datenbank durchführen zu können.**

## **Kontakt**

**CONZEPT 16-Server - Servicefunktionen im Hot-Standby Betrieb**

**Beschreibung der Servicefunktionen im Hot-Standby Betrieb**

**Im laufenden Betrieb können verschiedene Servicefunktionen an den Datenbanken durchgeführt werden. Die Funktionen werden über die Service-Seite der Web-Administration aufgerufen.**

## Kontakt

CONZEPT 16 - Datenbank & mehr X +

127.0.0.1:4745/area/service?ca1=CodeLibrary ... ⭐

### CONZEPT 16-Server

Status Datenbank Konfiguration

Manager Einstellungen Backup Hot-Standby Service Statistik

#### Service-Steuerung für "CodeLibrary"

##### Primärserver Status



Status	Master, Offen
IP-Adresse	192.168.0.1; 10.1.4.38
Zeitstempel	2017-12-01 11:05:06.1352653 (UTC)
Updatezähler	199.642
Aktive Sperren	

##### Sekundärserver Status



Status	Slave, Standby
IP-Adresse	192.168.0.2; 10.1.3.91
Zeitstempel	2017-12-01 11:05:06.1352653 (UTC)
Updatezähler	199.642
Aktive Sperren	

##### Benutzer

 Login sperren

 Login freigeben

 Alle Benutzer abmelden

 Upgrade sperren

 Upgrade freigeben

##### Hot-Standby

 Wechsel zu Primärsystem

 Wechsel zu Sekundärsystem

 Freigabe für Rollback

 Freigabe für Login

 Freigabe für Synchronisation

 Freigabe für Öffnen ohne Standby

 Datentransfer-Test starten

#### • Datenbankstatus

## Kontakt

Hier wird der Status beider Datenbanken angezeigt. In der ersten Zeile wird der Status der aktuellen Datenbank ausgegeben. Im normalen Betrieb sollte beim Primärsystem hier die Ausgabe "Master" stehen, gefolgt von dem Zustand der Datenbank. In der zweiten Zeile wird der Status der Datenbank auf dem anderen Server ausgegeben. Steht in der ersten Zeile der Status der Primärserver, wird in der zweiten Zeile "Sekundärer Server" ausgegeben, gefolgt vom Status der sekundären Datenbank. Folgende Einträge können für den Status auftreten:

◆ offen

Die Datenbank ist geöffnet. Dieser Zustand tritt ein, sobald sich ein Benutzer an der Datenbank anmeldet. Meldet sich der letzte Benutzer von der Datenbank ab, wird die Datenbank wieder geschlossen.

◆ geschlossen

Es sind keine Benutzer an der Datenbank angemeldet. Die Datenbank ist ordnungsgemäß geschlossen.

◆ Standby

Die Datenbank ist auf dem aktuellen Stand und bereit. Bei einem Ausfall des aktiven Servers kann diese Datenbank geöffnet werden.

◆ nicht abgeschlossen

Die Datenbank wurde nicht ordnungsgemäß geschlossen.

◆ Synchronisation

Die Standby-Datenbank wird gerade vom aktiven Server übertragen, damit beide Datenbanken auf dem gleichen Stand sind.

◆ unbekannt

Die Datenbank befindet sich nicht im angegebenen Verzeichnis oder es konnte keine Verbindung zum CONZEPT 16-Server auf dem anderen System hergestellt werden.

In der Web-Administration werden zusätzlich die Zeitstempel beider Datenbanken angezeigt. So kann auch ohne die Protokolldateien zu prüfen festgestellt werden, ob die Datenbanken synchron sind und welche der Datenbanken aktueller (die Master-Datenbank) ist.

• angemeldete Benutzer

Hier wird die Anzahl der an dieser Datenbank angemeldeten Benutzer ausgegeben.

• Login sperren

Wird diese Aktion durchgeführt, können sich keine weiteren Benutzer an die Datenbank anmelden. Das Anmelden neuer Benutzer kann durch die Aktion "Login freigeben" wieder zugelassen werden. Dieser Sperre wird nicht in die Datenbank eingetragen und somit beim Beenden des CONZEPT 16-Servers automatisch aufgehoben.

• Login freigeben

## Kontakt

Eine durch die Aktion "Login sperren" gesperrte Datenbank wird durch diese Aktion wieder zur Anmeldung durch neue Benutzer freigegeben.

- Alle Benutzer abmelden

Alle noch in der Datenbank angemeldeten Benutzer werden aus der Datenbank entfernt. Dabei werden alle noch offenen Transaktionen zurückgesetzt.

- Wechsel zu Primärsystem

Diese Aktion ist nur möglich, wenn beide Datenbanken (die des Primär- und die des Sekundärservers) geschlossen und synchron sind, wobei sich die aktive Datenbank auf dem Sekundärsystem befindet. Will sich nach dieser Aktion ein Benutzer an der Datenbank anmelden, wird die Datenbank vom Primärserver geöffnet.

- Wechsel zu Sekundärsystem

Diese Aktion ist nur möglich, wenn beide Datenbanken (die des Primär- und die des Sekundärservers) geschlossen und synchron sind, wobei sich die aktive Datenbank auf dem Primärsystem befindet. Will sich nach dieser Aktion ein Benutzer an der Datenbank anmelden, wird die Datenbank vom Sekundärserver geöffnet.

- Freigabe für Rollback

Wird eine Datenbank nicht ordnungsgemäß vom Server geschlossen, muss beim ersten Öffnen der Datenbank ein Rollback und eine anschließende Diagnose mit Recover durchgeführt werden (siehe auch Transaktionen und Datenbankdiagnose). Im Hot-Standby-Betrieb wird dies durch eine entsprechende Sperre (siehe Einrichten von Sperren bei beiden Datenbanken (aktiv und Standby) verhindert. Mit der "Freigabe für Rollback" wird diese Sperre entfernt und beim nächsten Öffnen der Datenbank wird dann automatisch ein Rollback und eine anschließende Diagnose mit Recover durchgeführt.



Ein automatisches Rollback wird nicht ohne das Aufheben der Sperre durchgeführt, da unter Umständen ein aktuellerer Stand der Datenbank auf dem anderen Server existiert und dieser eventuell gerade nicht verfügbar ist. Durch das Rollback würde in diesem Fall der Zeitstempel der älteren Datenbank aktualisiert und die aktuellere Datenbank bei wieder verfügbarem zweiten Server mit dem alten Stand überschrieben.

- Freigabe für Login

Diese Aktion entfernt die Sperre, welche durch eine nicht ausgewählte "Automatische Aktivierung" in der Datenbank gesetzt wurde. Danach können sich wieder Benutzer an der Datenbank anmelden. Diese Aktion kann nur bei geschlossener Datenbank durchgeführt werden.

- Freigabe für Synchronisation

Diese Aktion entfernt die Sperre, welche durch eine nicht ausgewählte "Automatische Synchronisation" in der Datenbank gesetzt wurde. Die Datenbank wird zur Synchronisation freigegeben und anschließend automatisch mit der aktuellen Datenbank synchronisiert. Nach einem Ausfall und der

## Kontakt

Wiederinbetriebnahme eines Servers muss diese Synchronisation durchgeführt werden, um die Daten, die in der Zwischenzeit in der aktuellen Datenbank verändert wurden, in die nun veraltete Datenbank zu übernehmen. Diese Aktion kann nur bei geschlossener Datenbank durchgeführt werden.

- Freigabe für Öffnen ohne Standby

Diese Aktion entfernt die Sperre Login sperren wenn Standby-Server nicht mehr verfügbar, welche in den Einstellungen der Datenbank gesetzt werden kann.

- Datentransfer-Test

Über diese Aktion kann die Verbindungsgeschwindigkeit zwischen dem Primär-und dem Sekundärserver überprüft werden. Der Test dauert ungefähr 10 Sekunden. Es wird danach die Übertragungsrate in KB pro Sekunde angezeigt.

## Kontakt

### CONZEPT 16-Server - Checkliste Hot-Standby Hot-Standby Checkliste zum Einrichten der Option und für den Fehlerfall Voraussetzungen

- Der Primärserver und der Sekundärserver verfügen über je zwei Netzwerkanschlüsse. Die Netzwerkanschlüsse, die für die Kommunikation zwischen den Servern zuständig sind, müssen mindestens eine Übertragungsrate von 1 GBit/s aufweisen.
- Die Kommunikation zwischen den beiden Servern erfolgt über eine exklusive Direktverbindung.
- Alle Netzwerkanschlüsse sind konfiguriert und das Protokoll TCP/IP ist installiert.
- Die Netzwerkanschlüsse für die direkte Verbindung sind in einem anderen Sub-Netz als die Netzwerkanschlüsse zum Firmennetzwerk.
- Zur eindeutigen Kennzeichnung des Primär- und des Sekundärservers sollten Markierungen an den entsprechenden Systemen angebracht werden.
- Die Hot-Standby Option dient zur Erhöhung der Verfügbarkeit einer oder mehrerer Datenbanken. Sie ist kein Ersatz für ein Datensicherungssystem.

### Installation der CONZEPT 16-Servers und der CONZEPT 16-Clients

- Die Programmstände und die Lizenz werden wie gewohnt eingerichtet. Zu beachten ist, dass auf beiden Servern der gleiche Programmstand und auch die gleiche Lizenzdatei c16.lic installiert sind.
- Bei der Verwendung der Lizenz mit Hardwareschutz, muss der Dongle-Treiber auf dem Primärserver installiert sein. Zusätzlich kann der Dongle-Treiber auch auf dem Sekundärserver installiert werden, damit dieser bei längerem Primärserverausfall auch auf das Sekundärsystem aufgesteckt werden kann.
- Bei der Verwendung der Lizenz mit Softwareschutz, darf die Lizenz-Identitätsdatei nur auf dem Primärserver vorhanden sein.
- Auf beiden Datenbankservern sollte die automatische Mailbenachrichtigung (siehe Konfiguration des Servers) eingerichtet werden, damit bei einem Ausfall der Systemadministrator sofort vom Wechsel zum Sekundärsystem benachrichtigt werden kann.
- Die Datenbank wird nur in das Datenbankverzeichnis des Primärservers kopiert.

### Eintragen der Datenbank beim Primärserver

Das Eintragen der Datenbank erfolgt über die Konfiguration des Servers.

- Im Dialog "Datenbank / Einstellungen" die Angaben für Name und Datenbank eintragen.

Name  
Datenbank

CodeLibrary  
c:\database\codelibrary

- Über den Link "Hot-Standby" die Parameter für die Hot-Standby Option aufrufen.

- In "Serveradresse" die IP-Adresse der Netzwerkkarte für die Direktverbindung und die IP-Adresse der normalen Netzwerkverbindung des Sekundärservers angeben.

## Kontakt

- Alle weiteren Einstellungen bleiben unverändert.

The screenshot shows the CONZEPT 16 software interface for managing databases. The title bar reads "CONZEPT 16 - Datenbank & mehr". The address bar shows the URL "127.0.0.1:4745/area/hotstandby?ca1=CodeLibrary". The main menu has tabs for Status, Datenbank (selected), Konfiguration, Manager, Einstellungen, Backup, Hot-Standby (selected), Service, and Statistik. A sub-menu titled "Hot-StandBy Einstellungen für 'CodeLibrary'" is open. It contains the following configuration options:

Serveradresse	192.168.0.2;10.1.3.91
Sekundärserver	<input type="checkbox"/>
Automatische Aktivierung	<input type="checkbox"/>
Automatische Synchronisation	<input checked="" type="checkbox"/>
Timeout Verbindungsaufbau [Sekunden]	3
Verbindungs-Intervall [Minuten]	5
Automatische Abschaltung bei Master-Master Betrieb	<input checked="" type="checkbox"/>
Login sperren wenn Standby-Server nicht mehr verfügbar	<input type="checkbox"/>

At the bottom right of the configuration panel is a green checkmark button labeled "Übernehmen" (Accept).

### Eintragen der Datenbank beim Sekundärserver

- Im Dialog "Datenbank / Einstellungen" die Angaben für Name und Datenbank eintragen. Der Name muss mit dem im Primärsystem angegebenen Aliasnamen übereinstimmen. Die physikalische Datenbankdatei auf dem Sekundärsystem sollte anders heißen, als die auf dem Primärsystem. Auf diese Weise kann eine einfache Unterscheidung zwischen den beiden Datenbanken getroffen werden. Zudem wird beim einfachen Kopieren zwischen den Servern eine der Datenbanken nicht aus Versehen überschrieben. Eine gute Unterscheidung ist durch die Erweiterung des Dateinamens mit -1 und -2 gegeben (Beispielsweise Codelibrary-1.ca1 und Codelibrary-2.ca1).

Name	Codelibrary
Datenbank	c:\db\codelibrary-2

- Über den Link "Hot-Standby" die Parameter für die Hot-Standby Option aufrufen.

## Kontakt

- In Serveradresse die IP-Adresse der Netzwerkkarte für die Direktverbindung und die IP-Adresse der normalen Netzwerkverbindung des Primärservers angeben.
- Die Kontrollkästchen für Sekundärsystem und Automatische Aktivierung aktivieren.
- Alle weiteren Einstellungen bleiben unverändert.

The screenshot shows the CONZEPT 16 software interface for managing databases. The title bar reads "CONZEPT 16 - Datenbank & mehr". The address bar shows the URL "127.0.0.1:4745/area/hotstandby?ca1=CodeLibrary". The main window is titled "CONZEPT 16-Server" and displays a navigation menu with tabs: Status, Datenbank, Konfiguration, Manager, Einstellungen, Backup, Hot-Standby (which is selected), Service, and Statistik. A sub-menu titled "Hot-StandBy Einstellungen für 'CodeLibrary'" is open, showing various configuration options. These include:

- Serveradresse: 192.168.0.1;10.1.4.38
- Sekundärserver: checked
- Automatische Aktivierung: checked
- Automatische Synchronisation: checked
- Timeout Verbindungsaufbau [Sekunden]: 3
- Verbindungs-Intervall [Minuten]: 5
- Automatische Abschaltung bei Master-Master Betrieb: checked
- Login sperren wenn Standby-Server nicht mehr verfügbar: unchecked

A green checkmark icon with the text "Übernehmen" is visible in the bottom right corner of the configuration area.

## Die Verknüpfung zum Starten der Datenbank

- In der Verknüpfung zum Öffnen der Datenbank werden anstelle von nur einem Server, beide Server in der Schreibweise Primärserver+Sekundärserver angegeben. Alle weiteren Parameter bleiben wie gewohnt.

Ziel: `winc.exe" 10.1.4.38+10.1.3.91 CodeLibrary`

## Überprüfen der Hot-Standby-Funktionalität

## Kontakt

- Vor der Überprüfung sollte unbedingt eine Datensicherung der Datenbank durchgeführt werden.
- Durch die erste Synchronisation soll nun die Datenbank auf den Sekundärserver übertragen werden. Die Synchronisation kann angestoßen werden, indem sich ein Benutzer an die Datenbank anmeldet. Die Datenbank wird daher nun mit Hilfe der angelegten Verknüpfung geöffnet.
- Im Hot-Standby Betrieb werden alle Meldungen des CONZEPT 16-Servers sowohl auf der Primärseite als auch auf der Sekundärseite in die Log-Datei der Datenbank geschrieben. In dieser Datei muss der Eintrag Synchronization started erzeugt worden sein (siehe [Log-Einträge](#)). Nach erfolgter Synchronisation erscheint hier Synchronization completed mit der benötigten Zeit in Sekunden. Die Datenbank befindet sich jetzt auch auf dem Sekundärserver. Wird die Datenbank anschließend auf dem Primärsystem geöffnet, ist ebenfalls der Eintrag Data area opened vorhanden.
- Den Client wieder beenden und abwarten bis die Datenbank geschlossen wurde. In der Log-Datei erscheint der Eintrag Data area closed.
- Jetzt kann der CONZEPT 16-Server auf dem Primärsystem beendet werden.
- Über die Verknüpfung die Datenbank erneut öffnen. Der Client sollte sich jetzt automatisch mit dem Sekundärserver verbinden.
- Um das System wieder in den ursprünglichen Zustand zu versetzen, müssen die Tätigkeiten ausgeführt werden, die unter "Was ist zu tun, wenn ... der Primärserver ausgefallen ist?" aufgeführt sind.

### Beispielezenarien im Hot-Standby Betrieb

Im Folgenden werden einige Szenarien im Hot-Standby Betrieb beschrieben und die jeweilige Lösungsmöglichkeit dazu erläutert. Die angegebenen Operationen sind im Client-Dialog "Service" erreichbar. Über diesen Dialog muss auch der Zustand des Systems überprüft werden, bevor die unten genannten Maßnahmen durchgeführt werden. Geschieht dies nicht, kann das einen Ausfall des Gesamtsystems zur Folge haben. Informationen über den Servicedialog können unter [Servicefunktionen im Hot-Standby Betrieb](#) entnommen werden.

-  Da beim Durchführen der Serviceaktionen in einigen Fällen keine Benutzer an der Datenbank angemeldet sein dürfen, können die Funktionen Login sperren, Login freigeben und Alle Benutzer abmelden verwendet werden. Bei letzterer Funktion ist darauf zu achten, dass alle offenen Transaktionen der angemeldeten Benutzer verworfen werden.
-  Alle durchzuführenden Freigaben, wie beispielsweise die Freigabe für Synchronisation, beziehen sich auf die empfohlene Hot-Standby Konfiguration (Siehe [Installation der Hot-Standby Option](#)).

## Kontakt

CONZEPT 16 - Datenbank & mehr X +

127.0.0.1:4745/area/service?ca1=CodeLibrary ... ⭐

### CONZEPT 16-Server

Status Datenbank Konfiguration

Manager Einstellungen Backup Hot-Standby Service Statistik

#### Service-Steuerung für "CodeLibrary"

#### Primärserver Status



Status Master, Offen  
IP-Adresse 192.168.0.1; 10.1.4.38

Zeitstempel 2017-12-01 11:05:06.1352653 (UTC)  
Updatezähler 199.642  
Aktive Sperren

#### Sekundärserver Status



Status Slave, Standby  
IP-Adresse 192.168.0.2; 10.1.3.91

Zeitstempel 2017-12-01 11:05:06.1352653 (UTC)  
Updatezähler 199.642  
Aktive Sperren

#### Benutzer

 Login sperren

 Login freigeben

 Alle Benutzer abmelden

 Upgrade sperren

 Upgrade freigeben

#### Hot-Standby

 Wechsel zu Primärsystem

 Wechsel zu Sekundärsystem

 Freigabe für Rollback

 Freigabe für Login

 Freigabe für Synchronisation

 Freigabe für Öffnen ohne Standby

 Datentransfer-Test starten

Was ist zu tun, wenn ...

## Kontakt

- ... der Primärserver gestoppt werden soll?
- ... der Primärserver wieder gestartet werden soll?
- ... der Sekundärserver gestoppt werden soll?
- ... der Sekundärserver wieder gestartet werden soll?
- ... der Primärserver ausgefallen ist?
- ... der Sekundärserver ausgefallen ist?
- ... beide Server ausgefallen sind?
- ... ein Wechsel vom Sekundär- zum Primärsystem erfolgen soll?
- ... ein Wechsel vom Primär- zum Sekundärsystem erfolgen soll?
- ... ein neuer Programmstand des Servers eingespielt werden soll?
- ... an der Datenbank Wartungsarbeiten durchgeführt werden sollen?
- ... ein Backup der Datenbank eingespielt werden soll?
- ... die Direktverbindung gestört ist?

*... der Primärserver gestoppt werden soll?*

- Der Zustand des Systems muss überprüft werden. Der Primärserver sollte die Datenbank als Master geöffnet haben. Der Sekundärserver als Slave. Hier kann ebenfalls überprüft werden, ob die Datenbank noch durch Benutzer geöffnet ist.
- Alle Benutzer müssen sich von der Datenbank abmelden.
- Auf dem Primärsystem wird die Funktion "Wechsel zum Sekundärsystem" aufgerufen.
- Das Primärsystem kann gestoppt werden.
- Falls das Primärsystem zu Wartungszwecken neu gestartet und wieder beendet wird, muss der automatische Start des CONZEPT 16-Servers temporär deaktiviert werden.
- Wird das Primärsystem über einen längeren Zeitraum nicht mehr gestartet und hat das Sekundärsystem keine eigene Lizenz, sollte der Dongle (bei Lizenz mit Hardwareschutz) auf das Sekundärsystem gesteckt werden. Der Dongle muss zum Öffnen der Datenbank auf dem Primärsystem wieder auf das Primärsystem gesteckt werden. Bei der Verwendung einer Lizenz mit Softwareschutz kann die entsprechende Lizenzidentitätsdatei (\*.idn) auf das Sekundärsystem kopiert werden, solange der Primärserver nicht gestartet wird. Wenn dieser wieder gestartet wird, ist die Lizenzidentitätsdatei auf dem Sekundärsystem zu entfernen, da sonst eine Mehrfachnutzung der Lizenz vorliegt.

*... der Primärserver wieder gestartet werden soll?*

- Zunächst wird der CONZEPT 16-Server auf dem Primärsystem wieder gestartet.
- Der Zeitstempel der Datenbanken muss überprüft werden. Der Zeitstempel wird auf der Seite "Service" der Web-Administration angezeigt. Die Datenbank mit dem älteren Zeitstempel wird bei der Synchronisation überschrieben. Die hier angegebenen Schritte dürfen nur durchgeführt werden, wenn die Datenbank auf dem Sekundärsystem einen aktuelleren Zeitstempel besitzt.
- Beim Primärsystem wird die Funktion "Freigabe für Synchronisation" aufgerufen, sofern die Synchronisations-Sperre gesetzt ist. Die Synchronisation wird durchgeführt, wenn die Datenbank auf dem Sekundärsystem wieder geöffnet wird. Ist die Datenbank offen, erfolgt die Synchronisation spätestens nach der als Verbindungs-Intervall definierten Zeit.
- Der Erfolg der Synchronisation muss anhand der Log-Datei des Datenbank-Prozesses überprüft werden.

## Kontakt

- Die Datenbank auf dem Sekundärsystem muss nun geschlossen werden.
- Auf dem Sekundärsystem wird die Funktion "Wechsel zum Primärsystem" aufgerufen.
- Auf dem Primärsystem wird die Funktion "Freigabe für Login" aufgerufen, sofern die Login-Sperre gesetzt ist.

*... der Sekundärserver gestoppt werden soll?*

- Vor dem Stoppen des Servers muss der Zustand des Systems überprüft werden. Die folgenden Maßnahmen können nur dann durchgeführt werden, wenn der Sekundärserver die Datenbank als Slave geöffnet hat.
- Das Sekundärsystem kann ohne weitere Vorbereitungen gestoppt werden.
- Falls das Sekundärsystem zu Wartungszwecken neu gestartet und wieder beendet wird, muss der automatische Start des CONZEPT 16-Servers temporär deaktiviert werden.
- Wird das Sekundärsystem über einen längeren Zeitraum nicht mehr gestartet, sollte die IP-Adresse des Sekundärservers aus Performance-Gründen in den Einstellungen der primären Datenbank entfernt werden. Wiederholte Versuche des Primärservers den Sekundärserver zu finden, können so unterbunden werden. Die Einstellungen müssen wieder geändert werden, wenn das Sekundärsystem wieder gestartet wird.

*... der Sekundärserver wieder gestartet werden soll?*

- Vor dem Starten des Datenbankservers sollte eine Sicherung der Datenbank angefertigt werden.
- Nach dem Start des Sekundärsystems erfolgt eine Synchronisation, sobald die Datenbank geöffnet oder ein Update-Ereignis durchgeführt wurde. Der Erfolg der Synchronisation muss anhand der Log-Datei des Datenbank-Prozesses überprüft werden.

*... der Primärserver ausgefallen ist?*

- Beim Ausfall des Primärservers werden alle CONZEPT 16-Clients mit der Meldung "Bereich: Kommunikation. Verbindung abgebrochen" beendet. Beim erneuten Starten arbeiten die Clients uneingeschränkt auf dem Sekundärsystem weiter. Zu diesem Zeitpunkt ist die Datenbank allerdings nicht durch einen zweiten Datenbankserver abgesichert.
- Auf dem Primärsystem muss der Grund für den Ausfall gefunden und beseitigt werden. Es muss in jedem Fall auch das Dateisystem des Primärsystems überprüft werden. Danach kann der CONZEPT 16-Server wieder gestartet werden.
- Es muss der Zustand des Systems (siehe Web-Administration) ermittelt werden. Der Status der Datenbank auf dem Primärserver ist "nicht abgeschlossen", auf dem Sekundärserver ist die Datenbank "offen" oder "geschlossen".
- Beim Primärsystem wird die Funktion "Freigabe für Synchronisation" aufgerufen, sofern die Synchronisations-Sperre gesetzt ist. Die Synchronisation wird durchgeführt, wenn die Datenbank auf dem Sekundärsystem wieder geöffnet wird. Ist die Datenbank offen, erfolgt die Synchronisation spätestens nach der als Verbindungs-Intervall definierten Zeit.

## Kontakt

- Der Erfolg der Synchronisation muss anhand der Log-Datei des Datenbank-Prozesses überprüft werden.
- Die Datenbank auf dem Sekundärsystem muss nun geschlossen werden.
- Auf dem Sekundärsystem wird die Funktion "Wechsel zum Primärsystem" aufgerufen.
- Auf dem Primärsystem wird die Funktion "Freigabe für Login" aufgerufen, sofern die Login-Sperre gesetzt ist.

*... der Sekundärserver ausgefallen ist?*

- Alle Clients können mit dem Primärserver weiterarbeiten. Zu diesem Zeitpunkt ist die Datenbank allerdings nicht durch einen zweiten Server abgesichert.
- Auf dem Sekundärsystem muss der Grund für den Ausfall gefunden und beseitigt werden. Es muss in jedem Fall auch das Dateisystem des Sekundärsystems überprüft werden. Vor dem Starten des CONZEPT 16-Server sollte eine Kopie der Datenbank angefertigt werden.
- Nach dem Start findet eine automatische Synchronisation statt. Danach ist die Datenbank wieder durch den zweiten Server abgesichert.
- Der Erfolg der Synchronisation muss anhand der Log-Datei des Datenbank-Prozesses überprüft werden.

*... beide Server ausgefallen sind?*

- Auf beiden Serversystemen muss der Grund für den Ausfall gefunden und beseitigt werden. Es müssen in jedem Fall auch die Dateisysteme der Server überprüft werden. Danach können die CONZEPT 16-Server wieder gestartet werden.
- Beide Datenbanken sollten nun inklusive Transaktionslog-Dateien und Datenbank-Log gesichert werden, damit in jedem Fall eine Sicherung der noch nicht identifizierten, aktuelleren Datenbank vorhanden ist.
- Es muss nun durch den Administrator festgestellt werden, welche der Datenbanken aktueller ist. Die Zeitstempel der Datenbanken werden in der Web-Administration angezeigt. Die Login-Einträge in den Log-Dateien oder der Zeitstempel der Transaktionslog-Dateien der Datenbanken können ebenfalls dazu verwendet werden. Danach muss für diese Datenbank die Hot-Standby Funktionalität deaktiviert (Serveradresse des anderen Servers in der Hot-Standby Konfiguration entfernen) und ein Rollback (siehe Transaktionen) durchgeführt werden.



**Kann die aktuellere Datenbank nicht bestimmt werden, muss für beide Datenbanken ein Rollback außerhalb des Hot-Standby Betriebs durchgeführt werden. Danach kann anhand des Datenbestandes die aktuellere Datenbank ermittelt werden.**

- Die aktuellere Datenbank wird unter dem korrekten Namen auf das Primärsystem verschoben. Die andere Datenbank wird gelöscht. Die Hot-Standby Funktionalität wird jetzt wieder aktiviert.
- Beim Öffnen der Datenbank wird eine Synchronisation vorgenommen und die Datenbank auf das Sekundärsystem übertragen. Die Datenbank ist jetzt wieder durch die Hot-Standby-Option abgesichert.

*... ein Wechsel vom Primär- zum Sekundärsystem erfolgen soll?*

## Kontakt

- Ein Wechsel vom Primär- zum Sekundärserver ist nur möglich, wenn beide Datenbanken synchron sind.
- Der Zustand des Systems sollte zunächst überprüft werden. Die folgenden Aktionen sollten nur durchgeführt werden, wenn die Datenbank auf dem Primärsystem als Master und auf dem Sekundärsystem als Slave geöffnet ist, bzw. die Datenbank auf dem Primärsystem einen aktuelleren Zeitstempel besitzt als auf dem Sekundärsystem.
- Von der Datenbank müssen sich alle Benutzer abmelden.
- Auf dem Primärserver muss für die betreffende Datenbank die Aktion "Wechsel zum Sekundärsystem" durchgeführt werden. Dadurch bekommt die Datenbank auf dem Sekundärserver einen neueren Zeitstempel, als die Datenbank auf dem Primärserver. Anschließend kann die Datenbank wieder geöffnet werden. Die Clients verbinden sich automatisch mit dem Sekundärserver.

*... ein Wechsel vom Sekundär- zum Primärsystem erfolgen soll?*

- Ein Wechsel vom Sekundär- zum Primärserver ist nur möglich, wenn beide Datenbanken synchron sind.
- Der Zustand des Systems sollte zunächst überprüft werden. Die folgenden Aktionen sollten nur durchgeführt werden, wenn die Datenbank auf dem Sekundärsystem als Master und auf dem Primärsystem als Slave geöffnet ist, bzw. die Datenbank auf dem Sekundärsystem einen aktuelleren Zeitstempel besitzt als auf dem Primärsystem.
- Von der Datenbank müssen sich alle Benutzer abmelden.
- Auf dem Sekundärserver muss für die betreffende Datenbank die Aktion "Wechsel zum Primärsystem" durchgeführt werden. Dadurch bekommt die Datenbank auf dem Primärserver einen neueren Zeitstempel, als die Datenbank auf dem Sekundärserver.
- Auf dem Primärserver muss für die betreffende Datenbank die Aktion "Freigabe für Login" durchgeführt werden, sofern die Login-Sperre gesetzt ist. Anschließend kann die Datenbank wieder geöffnet werden. Alle Clients verbinden sich dabei automatisch wieder mit dem Primärserver.

*... ein neuer Programmstand des Servers eingespielt werden soll?*

- Es müssen alle Benutzer alle Datenbanken verlassen.
- Der Zustand des Systems sollte zunächst überprüft werden. Die folgenden Aktionen gehen davon aus, dass die Datenbank auf dem Primärsystem als Master und auf dem Sekundärsystem als Slave geöffnet ist, bzw. die Datenbank auf dem Primärsystem einen aktuelleren Zeitstempel besitzt als auf dem Sekundärsystem. Ist die Datenbank auf dem Sekundärsystem die Masterdatenbank, dreht sich die Reihenfolge beim beenden der Server um.
- Als nächstes wird der CONZEPT 16-Server auf dem Sekundärsystem angehalten.
- Anschließend kann der CONZEPT 16-Server auf dem Primärsystem angehalten werden.
- Jetzt können die Programmstände auf dem Primär- und dem Sekundärserver aktualisiert werden. Die Programmstände und die Datei c16.lic müssen dabei auf beiden Systemen identisch sein.
- Als nächstes kann das Primärsystem wieder gestartet werden.
- Anschließend erfolgt der Start des Sekundärsystems.

## Kontakt

- Laufen beide Systeme, können sich wieder Benutzer an die Datenbanken anmelden.

*... ein Backup der Datenbank eingespielt werden soll?*

- Die Datenbank wird auf beiden Systemen gelöscht (Beziehungsweise umbenannt oder in ein anderes Verzeichnis verschoben).
- Die Datenbank aus dem Backup wird auf den Primärserver übertragen.
- Beim Anmelden des ersten Benutzers erfolgt die Synchronisation der Datenbank zum Sekundärserver.

*... an der Datenbank Wartungsarbeiten durchgeführt werden sollen?*

- Der Zustand des Systems sollte zunächst überprüft werden. Die folgenden Aktionen können nur durchgeführt werden, wenn die Datenbank auf dem Primärsystem als Master und auf dem Sekundärsystem als Slave geöffnet ist, bzw. die Datenbank auf dem Primärsystem einen aktuelleren Zeitstempel besitzt als auf dem Sekundärsystem.
- Das Vorgehen beim Update der Datenbank, bei der Durchführung einer Diagnose mit Recover oder einer Schlüsselanalyse und -reparatur ändert sich nicht durch den Einsatz der Hot-Standby-Option. Wichtig ist nur, dass alle Wartungsarbeiten ausschließlich auf der primären Seite durchgeführt werden können.
- Zunächst müssen alle Benutzer die Datenbank verlassen.
- Anschließend kann das Update eingelesen oder die Wartungsfunktion durchgeführt werden. Die Datenbank auf dem sekundären System wird nach der Wartung automatisch neu synchronisiert.

*... die Direktverbindung gestört ist?*

- Bei einer gestörten oder unzuverlässigen Direktverbindung ist kein korrekter Betrieb der Hot-Standby Option gewährleistet.
- Es muss das System bestimmt werden, welches als aktiver Datenbankserver arbeitet.
- Alle Benutzer müssen die Datenbank verlassen.
- Die Hot-Standby Funktionalität wird deaktiviert.
- Das Standby-System muss nun gestoppt werden.
- Die Datenbank auf dem Standby-System sollte umbenannt oder gelöscht werden.

## Kontakt

### Aufruf von Prozeduren beim Server

Verarbeitung von Prozeduren auf dem CONZEPT 16-Server.

Der CONZEPT 16-Server ist in der Lage A+ Prozeduren zu verarbeiten. Eine Prozedur wird mit dem Befehl RmtCall() auf dem Server gestartet. Die aufrufende Prozedur läuft dabei parallel zur aufgerufenen Prozedur.

Der Server kann nur einen beschränkten Befehlssatz interpretieren. Die ausführbaren Befehle sind in der Online-Hilfe mit dem Symbol  markiert.



Um eine Prozedur beim Server starten zu können, muss das Prozedurlimit (siehe Konfiguration der Datenbanken) größer 0 gesetzt sein.

## Kontakt

### Der CONZEPT 16-Druckertreiber

#### Beschreibung des Druckertreibers



Eine Auflistung der Betriebssysteme, auf denen der Druckertreiber betrieben werden kann, finden Sie in den [Systemvoraussetzungen](#).

Der CONZEPT 16-Druckertreiber ist ein Dienst, mit dem Druckjobs flexibel weiterverarbeitet werden können. Mit dem Druckertreiber werden verschiedene Drucker installiert. Wird auf einen dieser Drucker gedruckt, wird dieser Druckjob an eine CONZEPT 16-Prozedur weitergeleitet und kann dort auf einem anderen Drucker gedruckt, mit Hilfe des Acrobat Distillers oder von Ghostscript in ein PDF-Dokument oder in ein TIFF-Dokument umgewandelt werden. Dabei spielt es keine Rolle, welche Applikation auf den CONZEPT 16-Drucker gedruckt hat.

Der CONZEPT 16-Druckertreiber besteht aus folgenden Komponenten:

- Druckertreiber zur Erzeugung von Druckjobs im Postscript-Format
- Druckertreiber zur Erzeugung von Druckjobs im TIFF-Format
- Druckprozessor zur Verarbeitung der Druckjobs von CONZEPT 16-Druckertreibern

Bei der Installation auf einem Rechner werden standardmäßig alle Bestandteile lokal auf dem Rechner installiert. Sie können aber auch einzeln installiert werden. Die Installation wird im Abschnitt [Installation des CONZEPT 16-Druckertreibers](#) erläutert. Die Beschreibung der Funktionsweise befindet sich im Abschnitt [Druckprozessor - Funktionsweise](#).

## Kontakt

### Installation des Druckertreibers

#### Beschreibung der Installation des CONZEPT 16-Druckertreibers

Der Komponenten des CONZEPT 16-Druckertreibers werden über die CONZEPT 16 Installationsroutine eingerichtet. Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \Printer.

-  Bei der Installation des Druckprozessors wird die Druckwarteschlange des Betriebssystems gestoppt und anschließend wieder gestartet. Bestehen Abhängigkeiten zu anderen Diensten, kann die Druckwarteschlange unter Umständen nicht korrekt beendet oder neugestartet werden. In diesem Fall wird eine entsprechende Dialogbox ausgegeben. Die Abhängigkeiten der Druckerwarteschlange können dann manuell überprüft und die entsprechenden Dienste manuell angehalten und später wieder gestartet werden.

### Druckertreiber

Die Druckertreiber umfassen die zu installierenden Druckertreiber zum Generieren von PDF- und TIFF-Dokumenten und den sogenannten Port-Monitor, der für die Weiterleitung der Druckjobs an den Druckprozessor verantwortlich ist.

-  Die zu installierenden Druckertreiber können über die Konfigurationsdatei c16\_setup\_printer.cfg definiert werden, die vor dem Start der Installation in das Verzeichnis der Installationsroutine kopiert wird. Um bereits installierte Konfigurationsdateien zu überschreiben, muss die Installationsroutine mit dem Parameter /OVERWRITE angegeben werden. Für die Installation von Druckertreibern für den Einsatz als Netzwerkdrucker siehe Netzwerkdrucker.

#### Dateien der Druckertreiber

\driver	Dateien der Druckertreiber
c16_infsetup64.dll	
c16_prtdrv.inf	
c16_prtmoncfg.dll	
c16_prtmoncom32.dll	
c16_prtmoncom64.dll	
c16_prtmondrv32.dll	
c16_prtmondrv64.dll	
c16_prtpdf32.dll	
c16_prtpdf32.ini	
c16_prtpdf64.dll	
c16_prtpdf64.ini	
c16_prtpdf.ppd	
c16_prttif32.dll	
c16_prttif32.ini	
c16_prttif64.dll	
c16_prttif64.ini	
c16_prttif.gpd	

## Kontakt

\example

Beispielkonfigurationsdatei für die einzurichtenden  
Druckertreiber

c16\_setup\_printer.cfg

Druckprozessor

Der Druckprozessor führt die Verarbeitung der Druckjobs durch. Die

Installationsroutine richtet den Druckprozessor als Dienst ein.



Der Druckprozessor entnimmt seine Konfiguration beim Start des Dienstes aus der **Konfigurationsdatei des Druckprozessors**. Diese kann vordefiniert werden. Sie wird hierzu vor dem Start der Installation in das Verzeichnis der Installationsroutine kopiert.



Damit der Druckprozessor Druckaufträge von anderen Rechnern empfangen kann, muss der TCP/IP-Port 4729 auf dem Druckserver für eingehende Verbindungen geöffnet sein.

### Dateien des Druckprozessors

\service

Dateien des Druckprozessors

c16.tla

Token-Übersetzungstabellen

c16.vra

Ressourcendatei

c16\_bar2d\_w32.dll Barcode-Bibliothek (32-Bit)

c16\_chart\_w32.dll Chart-Bibliothek (32-Bit)

c16\_coded\_w32.dll CodeEdit-Bibliothek (32-Bit)

c16\_diff\_w32.dll Difference-Bibliothek (32-Bit)

c16\_graph\_w32.dll Graphics-Bibliothek (32-Bit)

c16\_icu\_w32.dll Unicode-Bibliothek (32-Bit)

c16\_obj\_w32.dll GUI-Objekte (32-Bit)

c16\_pdf\_w32.dll PDF-Bibliothek (32-Bit)

c16\_pfdx\_w32.dll PDF-Anzeige-Bibliothek (32-Bit)

c16\_ppcsvc.exe Druckprozessor

c16\_res\_w32.dll GUI-Ressourcen (32-Bit)

c16\_ssl\_w32.dll SSL-Bibliothek (32-Bit)

c16\_sys\_w32.dll System-Objekte (32-Bit)

c16\_xml\_w32.dll XML-Bibliothek (32-Bit)

c16\_zip\_w32.dll ZIP-Bibliothek (32-Bit)

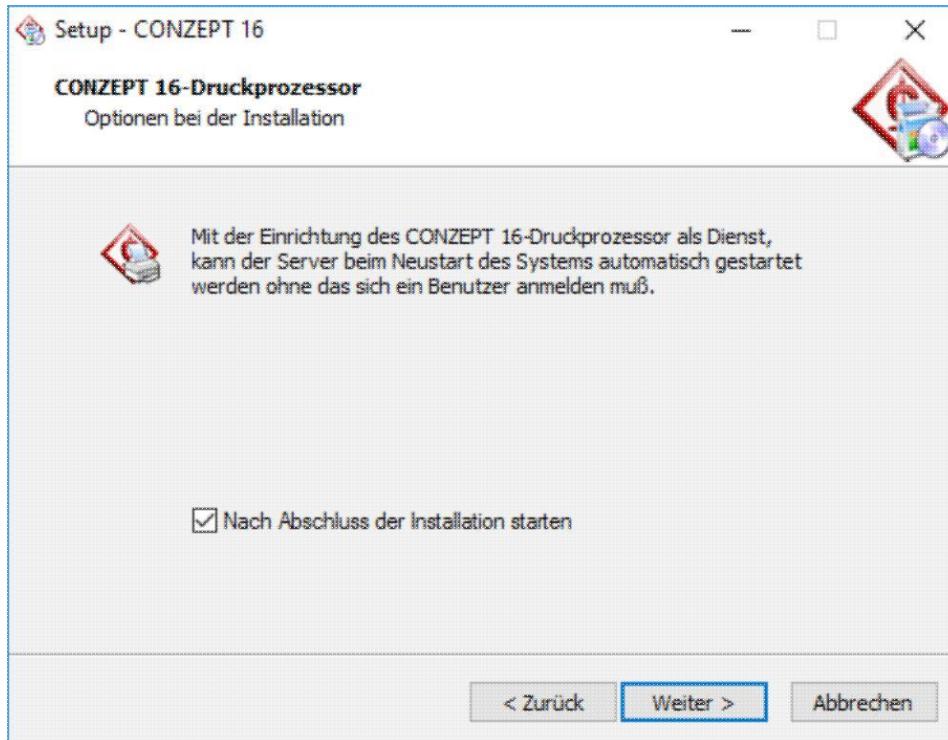
\example

c16\_ppcsvc.cfg Beispielkonfigurationsdatei des Druckerprozessors

Installationsverlauf

Der Druckprozessor kann nach dem Installationsvorgang automatisch gestartet werden. Über die Installationsroutine wird dies durch das Aktivieren der folgenden Option erreicht. Diese wird allerdings nur dann berücksichtigt, wenn eine Konfigurationsdatei für den Dienst vorhanden ist.

## Kontakt



**Der Druckprozessor ist in der Lage PDF-Dateien zu erzeugen. Dazu benötigt er entweder den Acrobat Distiller oder das frei verfügbare Ghostscript. Ist Ghostscript auf dem Zielsystem nicht eingerichtet, wird vor dem Start der Installation noch ein weiterer Dialog angezeigt. Hier kann der Download von Ghostscript gestartet werden.**

**Befindet sich die Ghostscript-Installationsdatei gs\*w\*\*\*.exe im gleichen Verzeichnis wie die Installationsroutine, kann mit dem Aktivieren der Option Ghostscript auch gleich mitinstalliert werden. In diesem Fall wird hinter der Option der Text "Ghostscript installieren" angezeigt. Die Installation von Ghostscript erfolgt unabhängig von der Installation des CONZEPT 16-Druckertreibers. Bei der Installation müssen alle Komponenten von Ghostscript installiert werden (entspricht der Standardeinstellung der Installationsroutine von Ghostscript).**

## Kontakt

### Konfigurationsdatei zur Installation des CONZEPT 16-Druckertreibers

#### Einstellungen bei der Installation

Bei der Installation des CONZEPT 16-Druckertreibers werden standardmäßig zwei Windows-Druckertreiber installiert. Einer der Druckertreiber erzeugt Druckjobs im Postscript-Format, welche dann zum Beispiel zu PDF-Dokumenten weiterverarbeitet werden können. Der andere erzeugt Druckjobs im TIFF-Format.

Sollen andere oder weitere Drucker eingerichtet werden, wird eine Konfigurationsdatei benötigt, die vor der Installation der Druckertreiberkomponenten in das Verzeichnis der Installationsroutine kopiert werden muss. Diese Konfigurationsdatei ist eine Text-Datei mit dem Namen **c16\_setup\_printer.cfg**, die mit jedem Text-Editor angelegt werden kann. Für jeden Windows-Drucker muss ein eigener Bereich angelegt werden, der bestimmt, ob ein PDF- oder ein TIFF-Dokument erzeugt wird. Innerhalb des Bereichs befinden sich die Einträge für den Druckertreiber.



Sind auf dem System bereits Druckertreiber installiert, befindet sich in dem Verzeichnis der Konfigurationsdateien bereits eine entsprechende Datei. Diese Datei wird beim Aufruf der Installationsroutine bevorzugt. Wird beim Starten der Installationsroutine die Option **/overwrite** angegeben, wird in jedem Fall die Datei im Verzeichnis der Installationsroutine verwendet.

- **c16\_printer\_name**

Name des Druckers. In Windows wird ein Drucker mit dem angegebenen Namen installiert.

- **c16\_printer\_port**

Dies ist der Port, auf den der Drucktreiber druckt. Der Port muss in der Form **C16\_PORT:\\<Rechner>\<Bereich>** angegeben werden. Als Rechner kann der Netzwerkname oder die IP-Adresse des Rechners angegeben werden, auf dem der Druckprozessor installiert ist. Läuft der Druckprozessor auf der gleichen Maschine, kann auch local angegeben werden.

Der Port definiert einen Abschnitt in der Konfigurationsdatei des Druckprozessors, der die notwendigen Informationen für die weitere Verarbeitung durch den Druckprozessor enthält.

Beispiel:

```
[PDF]c16_printer_name = "PDF-Printer" c16_printer_port = "C16_PORT:\\\\local\\\\default" [TIF]c16_printer_name = "TIFF-Printer" c16_printer_port = "C16_PORT:\\\\local\\\\default"
```

## Kontakt

### Konfigurationsdatei des Druckprozessors

#### Einstellungen des Druckprozessors

In der Konfigurationsdatei wird festgelegt, für welchen Druckerport welche Funktion aufgerufen wird. Bei der Installation des CONZEPT 16-Druckertreibers werden standardmäßig zwei Druckertreiber unter Windows mit installiert. Diese Druckertreiber drucken auf den Port \\local\\default. Der erste Name des Ports (local) definiert den Rechner, auf dem der Druckprozessor läuft, der zweite Name (default) definiert den Bereichsnamen, der sich in der Konfigurationsdatei des Druckprozessors wiederfindet.

Die Konfigurationsdatei (c16\_ppcsvc.cfg) ist eine Textdatei, die mit jedem Texteditor angepasst werden kann (siehe Beispiel-Datei). Sie ist in unterschiedliche Bereiche aufgeteilt, wobei jeder Bereich mit einem Bereichsnamen, in eckigen Klammern eingeschlossen, beginnt. Jeder Bereich umfasst folgende Einträge:

- **c16\_server**

Der Name oder die IP-Adresse des CONZEPT 16-Servers.

Beispiel:

```
c16_server = DbServer
```

- **c16\_database**

Name der Datenbank. Dies ist der symbolische Name, unter dem die Datenbank beim CONZEPT 16-Server eingetragen ist.

Beispiel:

```
c16_database = Order
```

- **c16\_user**

Benutzer der Datenbank. Der Druckprozessor meldet sich mit diesem Benutzernamen bei der Datenbank an. Der Benutzername muss in der Datenbank vorhanden und für den externen Zugriff freigegeben sein.

Beispiel:

```
c16_user = user
```

- **c16\_password**

Hier wird das Kennwort des Benutzers angegeben, der in Eintrag c16\_user eingetragen ist.

Hat der Benutzer kein Kennwort, muss "" angegeben werden.

Beispiel:

```
c16_password = ""
```

- **c16\_close\_delay (optional)**

Mit diesem Parameter wird eine Verzögerung in Sekunden eingestellt, die abgewartet wird, bis der Benutzer aus der Datenbank entfernt wird. Eine

## Kontakt

**Schließverzögerung ist dann sinnvoll, wenn mehrere Druckaufträge hintereinander ausgeführt werden. Zwischen den Druckjobs wird der Benutzer dann nicht aus der Datenbank abgemeldet.**

**Wird der Parameter nicht angegeben, hat er den Wert 60. Der Wert kann im Bereich von 0 (Benutzer wird sofort abgemeldet) und 86400 (Benutzer wird nach 24 Stunden abgemeldet) liegen.**

- **c16\_proc\_process**

**Hier wird die Prozedur oder die Funktion angegeben, die vom Druckprozessor aufgerufen werden soll. Die Angabe einer Funktion erfolgt in der Form <Prozedurname>:<Funktionsname>.**

**Beispiel:**

```
c16_proc_process = PrintProcessor:Bills
```



**Vom Druckprozessor können nur Prozeduren ausgeführt werden, die mit der Option A+ übersetzt wurden. Aufrufe der Anweisung CallOld() werden ignoriert.**

- **applogoptions (optional)**

**Mit dieser Option kann das Anwendungsprotokoll für den Druckprozessor aktiviert werden. Dies ist notwendig, damit bei Abstürzen Minidump-Dateien erzeugt werden. Zum Aktivieren muss der Wert auf 4 gesetzt werden. Zum Deaktivieren, muss 0 eingetragen, oder die Option entfernt werden.**

**Beispiel:**

```
applogoptions = 4
```

- **applogpath (optional)**

**Mit der Option wird der Pfad für das Anwendungsprotokoll festgelegt, sofern applogoptions gesetzt ist. Wird applogpath nicht definiert, wird das Anwendungsprotokoll im Verzeichnis der Konfigurationsdatei angelegt, bzw. geöffnet. Der Dateiname des Anwendungsprotokolles ist c16\_ppcdbg.lgb.**

**Beispiel:**

```
applogpath = C:\CONZEPT 16\Logs\
```

## Kontakt

### Netzwerkdrucker

#### Einrichten von CONZEPT 16-Druckertreibern als Netzwerkdrucker Homogene Systemarchitekturen

Die CONZEPT 16-Installationsroutine installiert die Druckertreiber jeweils passend zur lokalen Systemarchitektur. Das bedeutet, dass auf einem 32-Bit-System die 32-Bit-Druckertreiber und auf einem 64-Bit-System die 64-Bit-Druckertreiber von CONZEPT 16 eingerichtet werden. Sollen die Druckertreiber von einem 32-Bit-Druckserver an 32-Bit-Clients freigegeben werden, ist dies ohne weiteres möglich. Die Drucker werden über das Eigenschaftsmenü unter "Freigabe" freigegeben und auf den Clients entsprechend über die Druckerliste als neuer Netzwerkdrucker hinzugefügt. Gleiches gilt in einer homogenen Netzwerkumgebung mit 64-Bit-Systemen.

### Heterogene Systemarchitekturen

Die CONZEPT 16-Druckertreiber können ebenfalls in heterogenen Netzwerkumgebungen mit 32- und 64-Bit-Systemen eingesetzt werden. Dazu sind allerdings zuvor einige administrative Schritte notwendig. Ziel ist es, sowohl die 32-als auch die 64-Bit-Druckertreiber auf dem Druckserver einzurichten, damit alle Systemarchitekturen von dort aus bedient werden können. Dies ist zunächst nicht möglich, da beispielsweise auf einem 64-Bit-System keine 32-Bit-Drucker installiert werden können. Umgekehrt verhält es sich ebenfalls so. Um die im Beispiel benötigen 32-Bit-Systemressourcen auf das 64-Bit-System zu transferieren, sind die folgenden Schritte notwendig:

- **Schritt 1:** - Installieren des Druckertreiberpaketes auf dem 64-Bit-Druckserver
  - **Schritt 2:** - Manuelles Einrichten eines Netzwerkdruckers auf einem 32-Bit-System
  - **Schritt 3:** - Übertragen des Druckertreibers und der zugehörigen 32-Bit-Systemressourcen auf den 64-Bit-Druckserver
  - **Schritt 4:** - Netzwerkdrucker auf Clients beliebiger Systemarchitektur anlegen
- Schritt 1: Installieren des Druckertreiberpaketes auf dem 64-Bit-Druckserver

Im ersten Schritt müssen die Druckertreiber auf dem 64-Bit-Druckserver über die CONZEPT 16-Installationsroutine installiert werden. Bei der Installation verwendet CONZEPT 16 dabei ausschließlich die 64-Bit-Druckertreiber, da aufgrund der nicht vorhandenen 32-Bit-Systemressourcen die 32-Bit-Druckertreiber nicht angelegt werden können. Die Treiberdateien der 32-Bit-Treiber legt die Installationsroutine aber dennoch im Installationsverzeichnis ab. Die eingerichteten 64-Bit-Druckertreiber gibt der Administrator anschließend zur Nutzung im Netzwerk frei. Zur Vorbereitung für den zweiten Schritt wird zusätzlich das Installationsverzeichnis der Druckertreiber temporär für den Zugriff über das Netzwerk freigegeben (<Installationspfad CONZEPT 16>|Printer|Driver).

- Schritt 2: Manuelles Einrichten eines Netzwerkdruckers auf einem 32-Bit-System

## Kontakt

Auf einem beliebigen Druckclient wird nun der Drucker vom Druckserver als Netzwerkdrucker installiert. Während der Installation fordert der Druckclient die 32-Bit-Treiber für den Drucker an, da auf dem 64-Bit-Druckserver nur die 64-Bit-Ressourcen vorhanden sind. Im Auswahldialog für den 32-Bit-Druckertreiber wählt der Administrator das freigegebene Installationsverzeichnis auf dem Druckserver aus. Über die entsprechende Treiberdatei wird anschließend der Drucker mit den 32-Bit-Treibern auf dem Client eingerichtet (c16\_prtdrv.inf).



Beim Öffnen des freigegebenen Verzeichnisses, beziehungsweise beim Verbindungsauflauf zum Druckserver, muss darauf geachtet werden, dass Administratorenberechtigungen vorhanden sind. Diese sind notwendig um im nächsten Schritt vom Client aus die Treiberdateien auf den Druckserver zu übertragen.

- **Schritt 3: Übertragen des Druckertreibers und der zugehörigen 32-Bit-Systemressourcen auf den 64-Bit-Druckserver**

Um anderen 32-Bit-Clients ebenfalls die Drucker zur Verfügung zu stellen, müssen im dritten Schritt die 32-Bit-Druckerkomponenten mit den zugehörigen 32-Bit-Systemressourcen auf den 64-Bit-Druckserver übertragen werden. Hierzu wählt der Administrator auf dem Client den Netzwerkdrucker aus. Über die Eigenschaften wechselt er dann zum Reiterreiter "Freigabe". Unter "Zusätzliche Treiber" zeigt das System eine Liste der vom Druckserver verfügbaren Treiber für die verschiedenen Systemarchitekturen an. Hier kann der Administrator den entsprechenden Eintrag für 32-Bit-Systeme aktivieren. Der Client transferiert im Anschluss automatisch den eingerichteten Treiber und die benötigten 32-Bit-Systemressourcen auf den Druckserver.

- **Schritt 4: Netzwerkdrucker auf Clients beliebiger Systemarchitektur anlegen**

Auf dem 64-Bit-Druckserver befinden sich nun sowohl die 64-Bit-Druckertreiber als auch die 32-Bit-Druckertreiber inklusive der zugehörigen 32-Bit-Systemressourcen. Clients mit 64- und mit 32-Bit-Systemarchitekturen können nun vom Server bedient werden. Dazu legt der Administrator auf den Clients wie gewohnt neue Netzwerkdrucker an. Der Server liefert automatisch die Treiber für die vorhandene Architektur. Die Freigabe auf das Installationsverzeichnis der Druckertreiber kann nun wieder entfernt werden.

## Kontakt

### CONZEPT 16-Druckprozessor - Funktionsweise

#### Aufbereitung der Druckausgabe

Nach der Standard-Installation des CONZEPT 16-Druckertreibers sind auf dem entsprechenden Rechner zwei neue Druckertreiber installiert und der CONZEPT 16-Druckprozessor als Dienst gestartet. Der CONZEPT 16-Druckprozessor wird immer dann angesprochen, wenn eine Applikation (CONZEPT 16 oder andere Programme) auf einen CONZEPT 16-Drucker druckt. Der Druckjob wird an den Druckprozessor weitergeleitet und durch eine CONZEPT 16-Prozedur aufbereitet. Diese Aufbereitung kann eine oder mehrere der folgenden Funktionen umfassen:

- Drucken auf Windows-System oder -Netzwerkdrucker
- Erzeugen von PDF-Dokumenten
- Erzeugen von Multipage-TIFF-Dateien
- Erzeugen von Vorschaubildern in verschiedenen Formaten

Wird über die CONZEPT 16-Druckertreiber gedruckt, wird der Druckjob an den Druckprozessor weitergeleitet. Bei der Installation des CONZEPT 16 Druckertreibers werden standardmäßig zwei Druckertreiber installiert. Diese leiten ihre Druckjobs an einen lokal gestarteten Druckprozessor weiter. Wurden bei der Installation Druckerports angegeben, die auf einer anderen Maschine sind (siehe

c16\_setup\_printer.cfg), wird der Druckjob an einen Druckprozessor im Netz weitergeleitet. Der Druckprozessor verbindet sich mit einer CONZEPT 16-Datenbank und führt innerhalb dieser Datenbank eine Prozedur oder Funktion aus. Die Datenbank und die Funktion werden in einer Konfigurationsdatei angegeben.



**Der Druckprozessor wird als Dienst gestartet. Dienste laufen in einem anderen Benutzerkontext (meist SYSTEM), der andere Benutzerrechte als der angemeldete Benutzer besitzt. Auf diese Rechte gilt es besonders zu achten, wenn aus der Prozedur des Druckprozessors andere Drucker oder externe Dateien angesprochen werden.**

## Kontakt

**Die aufgerufene Funktion muss folgenden Funktionskopf besitzen:**

```
sub <Funktionsname>( aPpcObject : int; // Druckprozess-Objekt) : logic;
```

**Ist in der Konfigurationsdatei nur eine Prozedur eingetragen, muss die main-Funktion wie folgt definiert sein:**

```
main( aPpcObject : int; // Druckprozess-Objekt) : logic;
```

**Der Funktion des Druckprozessors wird das Objekt PpcObject übergeben. Innerhalb der Funktion wird mit den Ppc-Befehlen bestimmt, was aus dem Druckjob generiert werden soll. Im einfachsten Fall kann der Druckjob auf den Standarddrucker des Systems ausgedruckt werden.**

**Beispiel:**

```
sub PrintProcess( aPpcObject : int; // Druckprozess-Objekt) : logic;local{ tHdlPrtDevice : han
```

**Im folgenden noch ein Beispiel bei dem aus dem Druckjob ein PDF-Dokument und für die erste Seite ein Vorschaubild erzeugt wird:**

```
sub PrintProcess( aPpcObject : int; // Druckprozess-Objekt) : logic;{ // Vorschaubild erzeugen
```

**Die Befehle zum Generieren von Dateien oder Druckjobs geben einen Fehlerwert zurück. Konnte zum Beispiel ein PDF-Dokument nicht generiert werden, kann eine entsprechende Fehlerbehandlung in der Funktion des Druckprozessors durchgeführt werden.**

**Beispiel:**

## Kontakt

```
sub PrintProcess( aPpcObject : int; // Druckprozess-Objekt) : logic;local{ tErg : int;}{ // v
```

**Konnte ein Druckjob nicht verarbeitet werden, bleibt die Datei des Druckjobs im Temporärverzeichnis des Betriebssystems stehen. Beim nächsten Starten des Druckprozessors wird diese Datei erneut verarbeitet. Gibt die durch den Druckprozessor aufgerufene Funktion true zurück, wird die temporäre Datei gelöscht.**

**Beim Erzeugen einer PDF-Datei wird zunächst eine Postscript-Datei generiert, die dann weiter zu der PDF-Datei umgewandelt wird. Das Verzeichnis und der Name dieser Postscript-Datei kann über die Eigenschaft FileName des ppcObject ermittelt werden. Sollte diese Datei zu einem späteren Zeitpunkt oder zu Debug-Zwecken noch benötigt werden, kann sie innerhalb der Prozedur kopiert werden.**

## Kontakt

### Druckprozessor - Log-Datei

#### Einträge in die Log-Datei des Druckprozessors

Sobald der Druckprozessor gestartet wurde, legt er die Datei c16\_ppcsvc.log an. In dieser Datei werden alle Fehler- und sonstige Meldungen des Druckprozessors protokolliert. Sie kann mit einem einfachen Editor geöffnet werden und befindet sich im Standardverzeichnis für Konfigurationsdateien.

```
Date       Time      C  Message22.03.2006  09.47.09  i  Service start requested22.03.2006  09.47
```

Jeder Eintrag wird in eine eigene Zeile geschrieben. Dabei wird zunächst das Datum und die Uhrzeit geschrieben. Anschließend erfolgt die Fehlerklasse (C) und die Information im Klartext.

Die Meldungen werden in folgende Klassen eingeteilt:

- **Information (i)**

Dies sind rein informative Meldungen. Es liegt kein Fehler vor.

- **Startfehler (S)**

Dies sind Fehler beim Starten des Druckprozessors.

- **Error (E)**

Es ist ein Fehler aufgetreten. Die entsprechende Fehlermeldung wird dahinter angegeben.

Folgende Einträge können in der Datei enthalten sein:

#### Information

- **Service start requested**

Der Druckprozessor soll gestartet werden.

- **Service started**

Der Druckprozessor wurde gestartet.

- **Service start failed**

Der Druckprozessor konnte nicht gestartet werden. Weitere Informationen zu diesem Fehler befinden sich in der Ereignisanzeige des Betriebssystems.

- **Service stop requested**

Der Druckprozessor soll angehalten werden.

- **Service stopped**

Der Druckprozessor wurde angehalten.

- **Ghostscript not installed**

Auf dem System ist kein Ghostscript installiert.

- **Ghostscript path [...]**

Auf dem System ist Ghostscript installiert. Der Pfad der Installation wird in den

## Kontakt

eckigen Klammern angegeben.

### Startfehler

- Missing configuration file (...)

Es existiert keine Konfigurationsdatei im Verzeichnis des Dienstes.

- Port is not defined in CONZEPT 16-Print-Service configuration file [...] Der Bereichsnamen im Port des Druckers ist nicht in der Konfigurationsdatei des Druckprozessors enthalten.

- Section collision

In der Konfigurationsdatei des Druckprozessors sind zwei Bereiche mit dem gleichen Namen angegeben.

- [...] Configuration item '...' is too long

In der Konfigurationsdatei ist in dem in eckigen Klammern angegebenen Abschnitt der Wert des in Hochkomma angegebenen Eintrags zu lang.

### Error

- [...] Database connect : Server not found

Der in dem angegebenen Bereich der Konfigurationsdatei definierte Datenbankserver wurde nicht gefunden. Der Eintrag c16\_server muss überprüft werden.

- [...] Database connect : Database not found

Die in dem angegebenen Bereich der Konfigurationsdatei definierte Datenbank wurde nicht gefunden. Der Eintrag c16\_database muss überprüft werden.

- [...] Database connect : Database open failed

Die in dem angegebenen Bereich der Konfigurationsdatei definierte Datenbank konnte nicht geöffnet werden. Ursachen unter anderem eine falsche Datenbankversion sein oder die Datenbank ist bereits von einem anderen Programm geöffnet.

- [...] Database connect : Database is locked

Die in dem angegebenen Bereich der Konfigurationsdatei definierte Datenbank ist gesperrt.

- [...] Database connect : Database in exclusive use

Die in dem angegebenen Bereich der Konfigurationsdatei definierte Datenbank ist in exklusiver Benutzung (zum Beispiel durch eine Diagnose mit Recover).

- [...] Database connect : Invalid database type

Die in dem angegebenen Bereich der Konfigurationsdatei definierte Datenbank ist nicht kompatibel zur Version 3.5 oder höher.

## Kontakt

- [...] Database connect : User limit is reached

Das Benutzerlimit des in dem angegebenen Bereich der Konfigurationsdatei definierten Datenbankservers ist erreicht.

- [...] Database connect : Cannot start server

Der im angegebenen Bereich der Konfigurationsdatei definierte Datenbankserver konnte keinen Prozess zum Öffnen der Datenbank starten (zum Beispiel, weil der Server gerade heruntergefahren wird).

- [...] Database connect : User authorization failed

Die Anmeldung an die Datenbank ist mit den Informationen, die in dem in eckigen Klammern angegebenen Bereich der Konfigurationsdatei angegeben wurden nicht möglich. Die Eintragungen c16\_user und c16\_password müssen überprüft werden.

- [...] Database connect : Server release not supported

Die Version des im angegebenen Bereich der Konfigurationsdatei definierten Datenbankservers wird nicht unterstützt.

- [...] Database connect : Client upgrade missing

Die Lizenz des Servers im angegebenen Bereich der Konfigurationsdatei unterstützt nicht die gestartete Version des CONZEPT 16-Druckertreibers.

- [...] Database connect : Insufficient client license

Der Funktionsumfang der Serverlizenz beinhaltet nicht den CONZEPT 16-Druckertreiber. Der Druckprozessor kann die Datenbank daher nicht öffnen. Zum Betrieb des Druckertreibers wird eine Advanced- oder Enterprise-Edition benötigt.

- [...] Database connect : License communication error

Bei Herstellen des Kontakts zum im Bereich der Konfigurationsdatei angegebenem Server ist ein Fehler aufgetreten.

- [...] Database connect : License transmission error

Bei der Übertragung der Lizenz des im angegebenen Bereich der Konfigurationsdatei definierten Servers ist ein Fehler aufgetreten.

- [...] Database connect : License error occured

Bei der Übertragung der Lizenz des im angegebenen Bereich Konfigurationsdatei definierten Servers wurden keine korrekte Daten übertragen.

- [...] Database connect : License mismatch evaluation

Der im angegebenen Bereich der Konfigurationsdatei definierte Server ist ein Server mit einer Evaluierungslizenz.

- [...] Procedure call (...) [...]

Beim Aufruf der in dem angegebenen Bereich der Konfigurationsdatei definierten Funktion (c16\_proc\_process) ist ein Fehler aufgetreten. Die Fehlernummer und die aufgerufenen Funktion werden hinter der Information angegeben. In der Online-Hilfe kann über den Index aus der Fehlernummer, die

## Kontakt

Fehlerkonstante ermittelt werden.

- [...] Procedure with invalid resulttype [...]

Die in dem Bereich in der Konfigurationsdatei angegebene Funktion (c16\_proc\_process) liefert keinen logischen Ergebniswert zurück.

- Ghostscript processing failed

Bei der Verarbeitung des Druckjobs ist in Ghostscript ein Fehler aufgetreten.

- Print job is damaged [...]

Der Druckjob, der an den Druckprozessor übergeben wurde ist defekt.

- PpcMakePreview skipped [Page = ...] [Mode = ...]

Das Vorschaubild konnte nicht erzeugt werden. Die Anweisung wurde übergangen.

- PpcPrint failed (-9081)

Bei der Kommunikation mit dem Druckertreiber ist ein interner Fehler aufgetreten.

## Kontakt

### Externe Windows Programmierschnittstelle

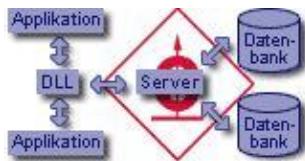
#### Beschreibung der Programmierschnittstelle



Eine Auflistung der Betriebssysteme, auf denen die Programmierschnittstelle betrieben werden kann, finden Sie in den [Systemvoraussetzungen](#).

Die CONZEPT 16-Programmierschnittstelle bietet dem Windows-Entwickler die Möglichkeit, aus einer selbst programmierten Anwendung heraus auf eine oder mehrere CONZEPT 16-Datenbanken zuzugreifen. Dabei handelt es sich um ein eigenständiges Produkt, welches als dynamische Funktionsbibliothek (DLL) für Windows realisiert ist.

Die DLL arbeitet im Client-Server-Betrieb in Verbindung mit einem CONZEPT 16-Server und unterstützt das Netzwerkprotokoll TCP/IP. Die Schnittstellendeklaration wird über eine Include-Datei eingebunden. In dieser Datei sind alle verfügbaren Typen, Konstanten und Funktionen definiert. Daneben ist eine entsprechende Import-Bibliothek vorhanden, die beim Linken des Programms mit eingebunden wird. Das erstellte Programm kann dann auf die Funktionen der DLL zugreifen. Die Schnittstelle kann gleichzeitig von beliebig vielen Programmen genutzt werden.



#### Arbeitsweise

Um auf eine Datenbank zuzugreifen, initialisiert das Programm zuerst die DLL. Anschließend wird eine sogenannte Instanz generiert, mit welcher sich eine Datenbank öffnen lässt. Da von einem Programm aus mehrere Instanzen angelegt werden können, ist auch der parallele Zugriff auf unterschiedliche Datenbanken möglich.

#### Funktionsumfang

Der Umfang und die Art der Funktionen sind eng an die Befehle des CONZEPT 16-Clients angelehnt. Daher ist der Einarbeitungsaufwand für einen CONZEPT 16-Entwickler sehr gering.

Außer dem vollständigen Zugriff auf die Datenstruktur- und Benutzerinformationen sind alle wesentlichen Funktionen für die Datenbearbeitung (Datensätze und Verknüpfungen lesen, Datensatzmanipulationen usw.) und den Zugriff auf interne Texte vorhanden. Beim Arbeiten mit mehreren geöffneten Datenbanken wird die Auswahl durch Angabe eines Handles für die jeweilige Instanz vorgenommen. Der Zugriff auf die Feldinhalte kann neben dem binären Datenformat auch mittels ASCII-Werten erfolgen.

#### Compiler

Die CONZEPT 16-Programmierschnittstelle wurde mit nachfolgenden Compilern getestet:

## Kontakt

- Borland C++
- Borland Delphi
- Microsoft Visual C++
- WATCOM C/C++

Andere Compiler, die oben nicht aufgeführt werden, sollten erst nach Rücksprache mit vectorsoft eingesetzt werden.



Die Programmierschnittstelle ist nicht thread-sicher. Aufrufe müssen selbstständig synchronisiert werden, wenn von unterschiedlichen Threads auf die gleiche Instanz zugegriffen wird.

Dieses Kapitel unterteilt sich in folgende Abschnitte:

- Installation der Programmierschnittstelle
- Funktionen
- Datenbankzugriffe
- Fehlerwerte

## Kontakt

## Installation der externen Windows Programmierschnittstelle

## **Beschreibung der Installation der externen Windows Programmierschnittstelle von CONZEPT 16**

Die externe Windows Programmierschnittstelle wird über die CONZEPT 16 Installationsroutine eingerichtet. Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \Api.

## Installationsverlauf

**Bei der Installation werden lediglich die entsprechenden Dateien in das Installationsverzeichnis kopiert.**

## Programmdateien

## **Dateien der Programmierschnittstelle**

c16.tla	Token-Übersetzungstabelle
c16.vra	Ressourcendatei
c16_bar2d_w32.dll	Barcode-Bibliothek (32-Bit)
c16_chart_w32.dll	Chart-Bibliothek (32-Bit)
c16_coded_w32.dll	CodeEdit-Bibliothek (32-Bit)
c16_diff_w32.dll	Difference-Bibliothek (32-Bit)
c16_graph_w32.dll	Graphics-Bibliothek (32-Bit)
c16_icu_w32.dll	Unicode-Bibliothek (32-Bit)
c16_icu_w64.dll	Unicode-Bibliothek (64-Bit)
c16_obj_w32.dll	GUI-Objekte (32-Bit)
c16_pdf_w32.dll	PDF-Bibliothek (32-Bit)
c16_pfdx_w32.dll	PDF-Anzeige-Bibliothek (32-Bit)
c16_pgx_w64.dll	Programmierschnittstelle (64-Bit)
c16_pgxe.dll	Erweiterte Programmierschnittstelle (32-Bit)
c16_pgxw.dll	Programmierschnittstelle (32-Bit)
c16_res_w32.dll	GUI-Ressourcen (32-Bit)
c16_ssl_w32.dll	SSL-Bibliothek (32-Bit)
c16_ssl_w64.dll	SSL-Bibliothek (64-Bit)
c16_sys_w32.dll	System-Objekte (32-Bit)
c16_xml_w32.dll	XML-Bibliothek (32-Bit)
c16_zip_w32.dll	ZIP-Bibliothek (32-Bit)

\example	<b>Beispiel in Delphi</b>
\include	<b>Header- und Unit-Dateien für C/C++ und Delphi</b>
c16.h	<b>Include-Datei für C/C++-Compiler</b>
c16_dll.h	<b>Include-Datei für C/C++-Compiler (eigene DLL)</b>
c16_pgx.h	<b>Include-Datei für C/C++-Compiler (c16_pgxw.dll / c16_pgx_w64.dll)</b>

## Kontakt

<b>\lib</b>	<b>Bibliotheken zum Linken der Programmierschnittstelle</b>
<b>c16_pgx_w64.lib</b>	<b>Import-Bibliothek zum Linken der Schnittstelle für 64-Bit in Microsoft Visual Studio</b>
<b>c16_pgxe.lib</b>	<b>Import-Bibliothek zum Linken der erweiterten Schnittstelle in Microsoft Visual Studio 2005 oder neuer</b>
<b>c16_pgxw.lib</b>	<b>Import-Bibliothek zum Linken der Schnittstelle in Microsoft Visual Studio 2005 oder neuer</b>

**Die erweiterte Programmierschnittstelle ermöglicht zusätzlich das Aufrufen von Oberflächenobjekten aus einer CONZEPT 16-Datenbank.**

## Kontakt

### Externe Windows Programmierschnittstelle - Funktionen Übersicht

#### über die Funktionen der Programmierschnittstelle

In der Programmierschnittstelle sind Funktionen für folgende Bereiche definiert:

- Argument-Funktionen
- Datenbank-Funktionen
- Datensatz-Funktionen
- Funktionen für binäre Objekte
- Funktionen für Selektionen
- Funktionen zum Ermitteln von Informationen
- Satzpuffer-Funktionen
- Sonstige Funktionen
- Text-Funktionen
- Transaktions-Funktionen
- Wartungs-Funktionen

Damit die Funktionen genutzt werden können, muss die Header-Datei c16\_pgx.h in das Programm eingebunden werden. Hier befinden sich notwendige Definitionen zur Benutzung der Schnittstelle.



Die Programmierschnittstelle ist nicht thread-sicher. Aufrufe müssen selbstständig synchronisiert werden, wenn von unterschiedlichen Threads auf die gleiche Instanz zugegriffen wird.

## Kontakt

c16\_pgx.h

# **Header-Datei der CONZEPT 16-Programmierschnittstelle**

Die Header-Datei bindet eine weitere Header-Datei (c16.h) mit allgemeinen Definitionen ein.

```
*****//  
***** open & close data  
***** SUBRECORD information  
C16API C16_FldInfoExt( const vPHANDLE aInstHdl, /* in: instance handle  
*****//  
***** LINK information  
/* read/write field data */C16API C16_FldData( const vPHANDLE aInstHdl,  
/* register external field buffer */C16API C16_RegExtFld( const vPHANDLE aInstHdl,  
*****// record information  
/* read linked record */C16API C16_RecLink( const vPHANDLE aInstHdl, /*in:  
/* copy record buffer - buffer handles are optional (0 if not used) */C16API C16_RecBufCopy( con  
/* read selection header */C16API C16_SelRead( const vPHANDLE aSelHdl, /*  
*****// text operations  
/* close text buffer */C16API C16_TextClose( const vPHANDLE aTextHdl /* in:  
/* write text data */C16API C16_TextWriteData( const vPHANDLE aTextHdl, /*  
*****// maintenance &  
/* query data area informations */C16API C16_QueryAreaInfo( const vPHANDLE aInstHdl,  
/* query diagnose results */C16API C16_DiagResults( const vPHANDLE aInstHdl,  
*****// BLOBs  
/* import BLOB from file */C16API C16_BinWriteFromFile( const vPHANDLE aBinHdl,  
*****// - compiler specific  
#pragma aux (dllstd) C16_RecRead#pragma aux (dllstd) C16_RecRead64#pragma aux (dllstd) C16_RecLin
```

## Kontakt

### c16.h

#### Header-Datei von CONZEPT 16

```
*****/*  
#ifdef _CHAR_UNSIGNEDtypedef char vCHAR; /* 8 bit unsigned  
// Structure of _TypeDecimaltypedef struct{ vBYTE length; vBYTE sign; vBYTEs exponent; vBYT  
*****/*  
#define C16ERR_NO SERVER CONNECTION -401#define C16ERR AREA NOT FOUND -402#define  
#define C16ERR_BIN_NAME_INVALID -851#define C16ERR_BIN_NO_PATH -852#define  
// compiler#define C16ERR_PCDC_CODE_OVERFLOW -1106#define C16ERR_PCDC_CONST_VAL  
*****/* - P-CODE INFORMATION  
#define C16ERR_LICENSE_SRVNOTSUPPORTED -2001#define C16ERR_LICENSE_CLNUPGRADE -2002#define  
#define _FileAttrOemMark 0x01#define _FileAttrTemp 0x02#define  
#define _RecFirst 0x00000001#define _RecLast 0x0000  
#define _RecCount 0#define _RecCountPos 5#define _RecC  
#define _SelFirst 0x01#define _SelLast 0x02#define  
#define _PgifType_Client 0x0001 /* Client-type */#define _PgifyTy  
#define _DiagRes_End 0#define _DiagRes_TreeCollision 1#define _Diag  
/* field information structure */typedef struct{ vINT InfoSize;  
/* key field information structure */typedef struct{ vINT InfoSize;  
/* text header information structure */typedef struct{ vINT InfoSize;  
/* interface information structure */typedef struct{ vINT InfoSize;  
/* user information structure */typedef struct{ vINT InfoSize;  
/* diagnose progress information structure */typedef struct{ vINT InfoSize;  
/* compiler information structure */typedef struct{ vERROR ErrorCode; vINT
```

## Kontakt

Externe Windows Programmierschnittstelle - Argument-Funktionen Übersicht über die Argument-Funktionen der Programmierschnittstelle

Diese Anweisungen können nur von DLL's genutzt werden, die mit dem CONZEPT 16-Befehl DllLoad() eingebunden wurden. Über diese Anweisungen können die Argumente bzw. Informationen zu den Argumenten ermittelt werden, die mit DllCall() übergeben wurden.

- C16 ArgCount
- C16 ArgInfo
- C16 ArgRead
- C16 ArgWrite

## Kontakt

**C16\_ArgCount(const vHANDLE aInstHdl,**

**vINT \* aArgCount) : vERROR**

Anzahl der übergebenen Argumente ermitteln.

aInstHdl Instanz-Handle

aArgCount Anzahl der übergebenen Argumente

Fehlerwert

C16ERR\_OK

kein Fehler

Resultat

**vERROR C16ERR\_INSTANCE\_HDL\_INVALID** Instanz-Handle

ungültig

Siehe [C16\\_ArgInfo\(\)](#), [C16\\_ArgRead\(\)](#), [C16\\_ArgWrite\(\)](#)

Mit dieser Funktion kann die Anzahl der Argumente ermittelt werden, die an die Einstiegsfunktion übergeben wurden. Die Einstiegsfunktion wird beim Laden der DLL ([DllLoad\(\)](#)) angegeben und mit ([DllCall\(\)](#)) in den CONZEPT 16-Funktionen aufgerufen.

In aArgCount muss ein Zeiger auf eine Variable vom Typ vINT übergeben werden. In dieser Variable steht nach dem Aufruf die Anzahl der übergebenen Variablen. Der Typ und die Übergabeart der Variablen kann mit der Funktion [C16\\_ArgInfo\(\)](#) ermittelt werden. Das Lesen des Wertes erfolgt dann über die Funktion [C16\\_ArgRead\(\)](#).

## Kontakt

**C16\_ArgInfo(const vHANDLE aInstHdl,  
const vINT aPosition,  
vINT\* aType,  
vINT\* aOptions,  
vINT\* aLength,  
vINT\* aMaxLength,  
vINT\* aElements) : vERROR**



Informationen zu einem Argument ermitteln

**aInstHdl** Instanz-Handle

**aPosition** Position des Parameters

**aType** Typ des Parameters

**aOptions** Art der Übergabe

**aLength** Länge des Arguments

**aMaxLength** Maximale Länge des Arguments

**aElements** Anzahl Elemente in einem Array

Fehlerwert

**C16ERR\_OK**

kein Fehler

**C16ERR\_INSTANCE\_HDL\_INVALID** Instanz-Handle

ungültig

**C16ERR\_NO\_ARGUMENT**

Das Argument

**Resultat** **vERROR**

ist nicht

vorhanden

**C16ERR\_ARGUMENT\_UNDEFINED** Das Argument

ist nicht

definiert /  
instanziert

Siehe [C16\\_ArgCount\(\)](#), [C16\\_ArgRead\(\)](#), [C16\\_ArgWrite\(\)](#)

Mit dieser Funktion können unterschiedliche Informationen zu einem Übergabeparameter ermittelt werden. In (aInstHdl) wird der Instanzen-Handle, in (aPosition) die Position des Argumentes angegeben. Alle weiteren Parameter sind Zeiger auf Variablen, die mit den entsprechenden Informationen gefüllt werden. Wird eine Information nicht benötigt, kann auch ein NULL-Zeiger übergeben werden.

- **aType**

Hier wird der Typ des übergebenen Arguments zurückgeliefert. Der Wert kann mit folgenden Konstanten verglichen werden:

Wert	Konstante	C-Typ
1	_TypeAlpha	vCHAR
2	_TypeDate	vDATE
4	_TypeWord	vWORD
5	_TypeDecimal	vBYTE[32]
6	_TypeMemo	vCHAR
7	_TypeInt	vLONGs
8	_TypeBigInt	vXLONGs
9	_TypeFloat	vFLOAT

## Kontakt

10 \_TypeLogic vBOOL8  
11 \_TypeTime vTIME

- aOptions

Ein Parameter kann sowohl als beschreibbar oder nicht-beschreibbar übergeben werden. Wird das Argument als beschreibbar übergeben, steht in dieser Variable der Wert \_ArgOptVAR (0x40). Wird eine Zeichenkette als beschreibbar übergeben, müssen die Parameter (aLength) und (aMaxLength) ebenfalls übergeben werden.

Wird ein Array übergeben, steht hier der Wert \_ArgOptARRAY (0x80). Die Anzahl der Elemente steht in (aElements).

- aLength und aMaxLength

Wird eine Zeichenkette übergeben, steht hier die Länge der Zeichenkette. Die maximale Länge, die diesem Wert zugewiesen werden kann, steht in (aMaxLength). Dieser Wert ist nur von Bedeutung, wenn die Zeichenkette als beschreibbar übergeben wurde.



Die Variable für die Zeichenkette muss mindestens ein Zeichen länger sein, damit das Nullzeichen Platz hat.

- aElements

Wurde ein Array übergeben, kann hier die Anzahl der Elemente ermittelt werden.

## Kontakt

**C16\_ArgRead(const vPHANDLE  
aInstHdl,  
const vINT aPosition,  
const vINT aIndex,  
void \* aData) : vERROR**



Argument lesen

aInstHdl Instanz-Handle

aPosition Position des Arguments

aIndex Element des Arrays

aData Übergebener Wert

Fehlerwert

**C16ERR\_OK**

kein Fehler

**C16ERR\_INSTANCE\_HDL\_INVALID** Instanz-Handle

ungültig

**C16ERR\_NO\_ARGUMENT**

Das Argument

**Resultat vERROR**

ist nicht

vorhanden

**C16ERR\_ARGUMENT\_UNDEFINED** Das Argument

ist nicht

definiert /  
instanziert

Siehe [C16\\_ArgCount\(\)](#), [C16\\_ArgInfo\(\)](#), [C16\\_ArgWrite\(\)](#)

Mit dieser Funktion kann ein Parameter, der beim Aufruf der Einstiegsfunktion mit [DllCall\(\)](#) übergeben wurde, ermittelt werden.

In (aInstHdl) wird die Instanz und in (aPosition) die Position des Parameters übergeben. Wurde ein Array übergeben, muss in (aIndex) das Element spezifiziert werden. Ob ein Array übergeben wurde und aus wie vielen Elementen es besteht kann mit der Funktion [C16\\_ArgInfo\(\)](#) ermittelt werden.

In (aData) muss ein Zeiger auf eine Variable mit einem korrekten Typ übergeben werden, in der der Wert zurückgegeben wird.



Wird eine Zeichenkette gelesen, muss im vCHAR-Array mindestens für ein Zeichen mehr Platz sein, als von [C16\\_ArgInfo\(\)](#) ermittelt wird, damit das abschließende Nullzeichen Platz hat.

## Kontakt

**C16\_ArgWrite(const vPHANDLE aInstHdl,  
const vINT aPosition,  
const vINT aIndex,  
void\* aData) : vERROR**



Argument schreiben

aInstHdl Instanz-Handle

aPosition Position des Arguments

aIndex Element des Arrays

aData Übergebener Wert

Fehlerwert

**C16ERR\_OK** kein Fehler

**C16ERR\_INSTANCE\_HDL\_INVALID** Instanz-Handle

ungültig

**C16ERR\_NO\_ARGUMENT** Das Argument

ist nicht

vorhanden

**Resultat vERROR**

**C16ERR\_ARGUMENT\_UNDEFINED** Das Argument

ist nicht  
definiert /  
instanziert

**C16ERR\_ARGUMENT\_READ\_ONLY** Das Argument

kann nicht  
beschrieben  
werden

Siehe [C16\\_ArgCount\(\)](#), [C16\\_ArgInfo\(\)](#), [C16\\_ArgRead\(\)](#)

Mit dieser Funktion kann ein Übergabeparameter der Funktion, die mit [DllCall\(\)](#) aufgerufen wurde und mit VAR deklariert ist, geschrieben werden.

Ob der entsprechende Parameter beschrieben werden kann, kann mit der Funktion [C16\\_ArgInfo\(\)](#) ermittelt werden.

In (aInstHdl) wird die Instanz übergeben. (aPosition) ist die Position des Arguments. Wurde an dieser Position ein Array übergeben, muss in (aIndex) das Element spezifiziert werden, das gesetzt werden soll. In (aData) wird der Wert angegeben, der an die aufrufende Funktion zurückgegeben werden soll.

Wird eine Zeichenkette zurückgegeben, die für die übergebene Variable zu lang ist, wird die Zeichenkette abgeschnitten.

## Kontakt

Externe Windows Programmierschnittstelle - Datenbank-Funktionen Übersicht über die Datenbank-Funktionen der Programmierschnittstelle

- [C16\\_CloseArea](#)
- [C16\\_InitInstance](#)
- [C16\\_InitPgif](#)
- [C16\\_OpenArea](#)
- [C16\\_TermInstance](#)
- [C16\\_TermPgif](#)

## Kontakt

**C16\_InitPgif(const vLONG  
aMemoryLimit,vPHANDLE\* aPgifHdl)  
: vERROR**  
Programmierschnittstelle initialisieren  
**aMemoryLimit** maximaler Speicherbereich  
Zeiger auf  
**aPgifHdl** Modul-Handle  
**Resultat** vERROR Fehlerwert  
**Siehe** [Befehle der Programmierschnittstelle.](#)  
[C16\\_TermPgif\(\)](#),  
[C16\\_InitInstance\(\)](#)

Der Befehl initialisiert die DLL und stellt den dazugehörigen Modul-Handle, der für das Anlegen der Datenbank-Instanzen benötigt wird, in (aPgifHdl) zur Verfügung.

Die maximale Größe des Speicherbereiches in Byte, der zum Laden der Datenstrukturen sowie zum Anlegen der Instanzen (jede Instanz belegt 64 KB) und der Datenpuffer angefordert werden kann, wird in dem Parameter (aMemoryLimit) angegeben. Hierbei handelt es sich um eine reine Speicherreservierung; der Speicher wird nicht gleich vom Programm belegt. Die benötigte Speichergröße hängt von der Anzahl und Größe der Datenstrukturen (ohne Masken) und der zusätzlichen Datenpuffer ab. In der Regel ist eine Größe von 4 MB ausreichend.

Der Parameter (aPgifHdl) beinhaltet nach erfolgreicher Ausführung des Befehls den Modul-Handle, der für das Anlegen der Datenbank-Instanz benötigt wird.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).



Die Funktion C16\_InitPgif() muss einmal zu Programmbeginn aufgerufen werden.  
Beispiel:

```
void main(void){    char pszMsgText[80];    vERROR nErg;                // Funktionsergebnis    vPHANDLE hModul
```

## Kontakt

**C16\_InitInstance**(const  
vHANDLE  
aPgifHdl,vHANDLE\* aInstHdl)  
: vERROR  
Datenbank Instanz initialisieren  
aPgifHdl Modul-Handle  
aInstHdl Instanz-Handle  
Resultat vERROR Fehlerwert

Siehe  
Befehle der Programmierschnittstelle,

C16\_InitPgif(),  
C16\_TermInstance()

Der Befehl initialisiert eine Datenbank-Instanz und stellt den dazugehörigen Instanz-Handle, der für das Öffnen einer Datenbank benötigt wird, in (aInstHdl). In (aPgifHdl) wird der von C16\_InitPgif() bereitgestellte Modul-Handle übergeben.



Die Programmierschnittstelle ist nicht thread-sicher. Aufrufe müssen selbstständig synchronisiert werden, wenn von unterschiedlichen Threads auf die gleiche Instanz zugegriffen wird.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

Für jede Datenbank, auf die zugegriffen wird, ist eine eigene Datenbank-Instanz anzulegen.

**Beispiel:**

```
void main(void){  char pszMsgText[80];  vERROR nRes;          // Funktionsergebnis  vHANDLE hModul
```

## Kontakt

```
C16_OpenArea(const  
vHANDLE aInstHdl,  
const vCHAR* aServerName,  
const vCHAR*  
aServerPassword,  
const vCHAR* aAreaName,  
const vCHAR* aUserName,  
const vCHAR* aUserPassword,  
const vCHAR* aReserved,  
const vCHAR* aLocalDSPPath,  
const vINT aProcCacheKB) :  
vERROR  
Datenbank öffnen  
aInstHdl      Instanz-Handle  
aServerName   Protokoll und Name  
              des Servers  
              Kennwort des  
aServerPassword Servers  
aAreaName     Pfad der Datenbank  
              (ohne .CA1)  
aUserName     Benutzer  
              Kennwort des  
aUserPassword Benutzers  
aReserved     Reserviert  
aLocalDSPPath Pfad für lokale  
              Datenstruktur  
              Größe des  
aProcCacheKB  Prozedurcaches  
  
Resultat       vERROR Fehlerwert  
              Befehle der  
              Programmierschnittstelle.  
Siehe          C16_CloseArea(),  
              C16_InitInstance()
```

Mit diesem Befehl wird eine Verbindung zum Server aufgebaut, die angegebene Datenbank geöffnet und ihre Datenstruktur in den Hauptspeicher geladen.

Im Parameter (aInstHdl) ist der von C16\_InitInstance() bereitgestellte Instanz-Handle zu übergeben. Die Anmeldung an den Server geschieht über die Parameter (aServerName) und (aServerPassword). Der Servername setzt sich aus dem Protokoll ("TCP") und dem Servernamen zusammen (TCP:<Servername>).

Wurde keine Host-Tabelle angelegt, kann sich eine externe Programmierschnittstelle, nur mit diesem Server verbinden. Die Host-Tabelle wird im gleichen Verzeichnis, in dem sich die Datei C16\_PGXW.DLL befindet, angelegt. Bei der Tabelle handelt es sich um eine Text-Datei mit dem Namen C16\_PGXW.HST. In dieser Datei wird das Protokoll und der Name oder die IP-Adresse des Servers angegeben, auf den die Schnittstelle geschützt ist.

**Beispiel:**

## Kontakt

TCP: 10.0.0.1

Der Parameter (aServerPassword) beinhaltet einen Zeiger auf das Passwort der Datenbank. Dieses Passwort kann über die Schaltfläche [Kennwort] im Dialog "Kennwortabfrage" angegeben werden. Die Angabe ist nur notwendig, wenn die Datenbank über das Passwort geschützt worden ist. Ist dies nicht der Fall bzw. ist die Datenbank nicht passwortgeschützt, wird NULL übergeben.

(aAreaName) enthält einen Zeiger auf eine Zeichenkette, in welcher der komplette Pfad der Datenbank angegeben wird (der Datenbankname ist ohne .CA1-Erweiterung anzugeben). Dies ist der gleiche Pfad, der auch zur Anmeldung der Datenbank beim Server angegeben wurde. Alternativ kann hier auch der symbolische Name der Datenbank angegeben werden.

Bei der Verwendung der **Hot-Standby**-Option des Servers muss der symbolische Name angegeben werden. In dem Übergabeparameter (aServerName) werden beide Server (der Primär- und der Sekundärserver) angegeben. Die beiden Server werden durch "+" voneinander getrennt (<Protokoll>:<Servername>+<Servername>).

Zur Anmeldung in der Datenbank wird ein Benutzer und das zugehörige Passwort benötigt. Der Benutzer wird in (aUserName) (Zeiger auf eine String-Konstante oder -Variable) übergeben. Das Passwort steht in (aUserPassword) (Zeiger auf eine String-Konstante oder -Variable). Hat der Benutzer kein Passwort, wird NULL übergeben.

 Der angegebene Benutzer (aUserName) benötigt für einen Zugriff auf die Datenbank eine entsprechende Berechtigung. Dazu muss die Option "Externer Zugriff" in den Programmrechten des Benutzers gesetzt sein.  
Im Parameter (aReserved) muss NULL übergeben werden.

Mit dem Parameter (aLocalDSName) wird ein Zeiger auf eine Zeichenkette, in welcher der Pfad der lokalen Datenstruktur, angegeben wird, übergeben (nur in Verbindung mit der WAN-Option des Servers). Ist keine lokale Datenstruktur vorhanden, wird NULL übergeben.

Der Parameter (aProcCacheKB) enthält die Größe des Prozedurpuffers für die Datenbank in Kilobyte. Der Puffer muss größer 0 sein, wenn eine CONZEPT 16-Prozedur ausgeführt werden soll. Alternativ kann die Option \_OpenMode5x angegeben werden, um Datenbanken zu öffnen, deren Stand kleiner als 5.8 ist.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

Es können mehrere Datenbanken gleichzeitig geöffnet sein. Für jede geöffnete Datenbank muss eine eigene Datenbank-Instanz initialisiert werden (siehe [C16\\_InitInstance\(\)](#)).

**Beispiel:**

```
void main(void){    char pszMsgText[80];    vERROR nErg;           // Funktionsergebnis    vHANDLE hModul
```

## Kontakt

**C16\_CloseArea(const  
vHANDLE aInstHdl) : vERROR**  
Datenbank schließen  
aInstHdl Instanz-Handle  
Resultat vERROR Fehlerwert

Siehe

Befehle der  
Programmierschnittstelle,  
C16\_OpenArea(),  
C16\_TermInstance()

Diese Funktion schließt eine mit C16\_OpenArea() geöffnete Datenbank.

In dem Parameter aInstHdl wird der von C16\_InitInstance() bereitgestellte Instanz-Handle angegeben, der beim Öffnen der Datenbank mit C16\_OpenArea angegeben wurde.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).



Diese Funktion muss vor Programmende für jede geöffnete Datenbank aufgerufen werden.

Beispiel:

```
void main(void){  char pszMsgText[80];  vERROR nRes;          // Funktionsergebnis  vHANDLE hModul
```

## Kontakt

**C16\_TermInstance(vPHANDLE\***

**aInstHdl) : vERROR**

Datenbank-Instanz beenden

aInstHdl Instanz-Handle

Resultat vERROR Fehlerwert

Siehe Befehle der Programmierschnittstelle,

C16\_InitInstance(),  
C16\_CloseArea()

Der Befehl beendet eine Datenbank-Instanz. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).



Diese Funktion muss vor Programmende für jede aktive Datenbank-Instanz aufgerufen werden.

Beispiel:

```
void main(void){  char pszMsgText[80];  vERROR nRes;          // Funktionsergebnis  vPHANDLE hModul
```

## Kontakt

**C16\_TermPgif(const vPHANDLE  
aPgifHdl) : vERROR**  
Programmierschnittstelle beenden  
aPgifHdl Zeiger auf Modul-Handle  
Resultat vERROR Fehlerwert

Siehe [Befehle der  
Programmierschnittstelle](#),

[C16\\_InitPgif\(\)](#),  
[C16\\_TermInstance\(\)](#)

Der Befehl beendet das Programmierschnittstellen-Modul und gibt den reservierten Speicher frei.

In (aPgifHdl) steht der von [C16\\_InitPgif\(\)](#) bereitgestellte Modul-Handle.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).



Diese Funktion muss einmal vor Programmende aufgerufen werden.

Beispiel:

```
void main(void){    char pszMsgText[80];    vERROR nRes;                // Funktionsergebnis    vPHANDLE hModul
```

## Kontakt

Externe Windows Programmierschnittstelle - Datensatz-Funktionen Übersicht über  
die Datensatz-Funktionen der Programmierschnittstelle

- [C16 ExcFileData](#)
- [C16 ExcFldData](#)
- [C16 ExcSbrData](#)
- [C16 FldData](#)
- [C16 FldDataChar](#)
- [C16 FldDataRaw](#)
- [C16 RecDelete](#)
- [C16 RecDeleteAll](#)
- [C16 RecInsert](#)
- [C16 RecLink](#)
- [C16 RecRead](#)
- [C16 RecRead64](#)
- [C16 RecReplace](#)
- [C16 SbrSetStatus](#)
- [C16 SetExcMode](#)

## Kontakt

**C16\_ExcFileData(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vHANDLE aBuffHdl,  
const vBOOL aWriteFlag) :**  
**vERROR**

Datensatz übertragen

**aInstHdl** Instanz-Handle

**aFileNo** Dateinummer

**aBufHdl** Datensatzpuffer

**aWriteFlag** Datensatz schreiben  
(true), lesen (false)

**Resultat** vERROR Fehlerwert

Befehle der  
Programmierschnittstelle,  
**Siehe** C16\_ExcFldData(),  
C16\_ExcSbrData(),  
C16\_RegExtFld()

Veranlasst den Transfer von Daten zwischen allen registrierten Programmvariablen (vgl. [C16\\_RegExtFld\(\)](#) und den mit ihnen verbundenen Feldern im Feldpuffer der Datei bzw. einem mit [C16\\_RecBufCreate\(\)](#) erzeugten Puffer.

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben.

In (aFileNo) wird die Dateinummer angegeben, deren Feldpuffer gelesen oder geschrieben werden sollen. Soll anstelle des Feldpuffers der Datei ein mit [C16\\_RecBufCreate\(\)](#) erzeugter Puffer verwendet werden, ist in (aBuffHdl) der Puffer-Handle anzugeben, sonst NULL. Der Parameter (aWriteFlag) gibt an, ob die in den registrierten Variablen bereitgestellten Daten in die Puffer von (aFileNo) geschrieben werden (true) oder der Inhalt des Datensatzpuffers von (aFileNo) in die registrierten Variablen kopiert werden (false).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_ExcFldData(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vINT aFldNo,  
const vBOOL aWriteFlag) :**

**vERROR**

Feld übertragen

aInstHdl	Instanz-Handle
aFileNo	Dateinummer
aSbrNo	Teildatensatznummer
aFldNo	Feldnummer
	Feld schreiben
aWriteFlag	(true), lesen (false)

### Befehle der

Programmierschnittstelle,  
C16\_ExcFileData(),

Siehe

C16\_ExcSbrData(),  
C16\_FldData(),  
C16\_RegExtFld()

Veranlasst den Transfer von Daten zwischen einer registrierten Programmvariable (vgl. C16\_RegExtFld()) und dem mit ihr verbundenen Feld im Feldpuffer der Datei.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Mit (aFileNo), (aSbrNo) und (aFldNo) wird das Feld identifiziert. Der Parameter (aWriteFlag) gibt an, ob das in der registrierten Variablen bereitgestellte Datum in die Puffer von (aFileNo) geschrieben werden (true) oder der Inhalt des Datensatzpuffers von (aFileNo) in die registrierte Variable kopiert wird (false).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_ExcSbrData(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vBOOL aWriteFlag) :**  
**vERROR**  
**Teildatensatz übertragen**  
**aInstHdl** Instanz-Handle  
**aFileNo** Dateinummer  
**aSbrNo** Teildatensatznummer  
**Teildatensatz**  
**aWriteFlag** schreiben (true),  
lesen (false)  
**Resultat** **vERROR Fehlerwert**  
**Befehle der**  
**Programmierschnittstelle,**  
**C16\_ExcFileData(),**  
**C16\_ExcFldData(),**  
**C16\_FldData(),**  
**C16\_RegExtFld()**

Siehe  
Veranlasst den Transfer von Daten zwischen allen registrierten Programmvariablen (vgl.  
**C16\_RegExtFld()** und den mit ihr verbundenen Feldern des angegebenen Teildatensatzes im  
Feldpuffer der Datei.

Im Parameter (**aInstHdl**) wird der von **C16\_InitInstance()** bereitgestellte Instanz-Handle, der in  
der Funktion **C16\_OpenArea()** zum Öffnen der Datenbank verwendet wurde, übergeben.

Mit (**aFileNo**) und (**aSbrNo**) wird die Dateinummer und die Teildatensatznummer angegeben, deren  
Feldpuffer gelesen oder geschrieben werden sollen. Es werden nur die Feldpuffer bzw. Variablen  
übertragen, die registriert (**C16\_RegExtFld()**) und in dem Teildatensatz enthalten sind. Der Parameter  
(**aWriteFlag**) gibt an, ob das in der registrierten Variablen bereitgestellten Datum in die Puffer von  
(**aFileNo**) geschrieben werden (true) oder der Inhalt des Datensatzpuffers von (**aFileNo**) in die  
registrierten Variablen kopiert wird (false).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen  
Fehlerwert (siehe **Fehlerwerte**).

## Kontakt

```
C16_FldData(const vHANDLE  
aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vINT aFldNo,  
void* aFldBuf,  
const vBOOL aWriteFlag) :  
vERROR
```

Feldinhalte übertragen

aInstHdl	Instanz-Handle
aFileNo	Dateinummer
aSbrNo	Teildatensatznummer
aFldNo	Feldnummer Zeiger auf
aFldBuf	Feldpuffer Feld schreiben
aWriteFlag	(true), lesen (false)

### Befehle der

#### Programmierschnittstelle,

Siehe [C16\\_ExcFldData\(\)](#),  
[C16\\_FldDataChar\(\)](#),  
[C16\\_FldDataRaw\(\)](#)

Liest den Inhalt eines Feldes in einen Puffer bzw. schreibt den Inhalt eines Puffers in ein Feld. Daten werden binär - also Byte für Byte - ausgelesen bzw. geschrieben, eine Typprüfung findet nicht statt. Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion

[C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. Die Parameter (aFileNo), (aSbrNo) und (aFldBuf) identifizieren eindeutig das Feld, das ausgelesen bzw. geschrieben werden soll. Der Parameter (aWriteFlag) gibt an, ob die in (aFldBuf) bereitgestellten Daten in den Puffer von (aFldNo) geschrieben werden (true) oder der Inhalt des Feldpuffers (aFldNo) nach (aFldBuf) kopiert wird (false).

Der Rückgabewert vom Typ [vERROR](#) beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_FldDataChar(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vINT aFldNo,  
vCHAR\* aFldBuf,  
const vINT aFormatFlags,  
const vINT aPostcomma,  
const vBOOL aWriteFlag) :**  
**vERROR**

Feld übertragen und konvertieren

<b>aInstHdl</b>	Instanz-Handle
<b>aFileNo</b>	Dateinummer
<b>aSbrNo</b>	Teildatensatznummer
<b>aFldNo</b>	Feldnummer
<b>aFldBuf</b>	Zeiger auf Feldpuffer
<b>aFormatFlags</b>	Formatierung
<b>aPostcomma</b>	Nachkommastellen
<b>aWriteFlag</b>	Feld schreiben (true), lesen (false)
<b>Resultat</b>	vERROR Fehlerwert <u>Befehle der Programmierschnittstelle,</u> <u>C16_ExcFldData()</u> ,
<b>Siehe</b>	<u>C16_FldData()</u> , <u>C16_FldDataRaw()</u> , <u>C16_SetCharDefs()</u>

Liest den Inhalt eines Feldes und konvertiert ihn in eine ASCII-Zeichenkette bzw. konvertiert eine ASCII-Zeichenkette in das entsprechende Datenformat und schreibt sie in das angegebene Feld. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Parameter (aFileNo), (aSbrNo) und (aFldBuf) identifizieren eindeutig das Feld, das ausgelesen bzw. geschrieben werden soll. Die Art der Formatierung wird in den Parametern (aFormatFlags) und (aPostcomma) angegeben. In (aFormatFlags) kann eine oder mehrere (logische OR-Verknüpfung) der nachfolgenden symbolischen Konstanten angegeben werden:

<b>_FmtNone</b>	keine bestimmte Formatierung
<b>_FmtTSep</b>	Tausendertrennung
<b>_FmtZeroSupr</b>	Nullunterdrückung
<b>_FmtPlusSign</b>	Positive Werte werden mit einem vorangestellten Plussymbol versehen
<b>_FmtDateLongYear</b>	Jahr bei Datumsfeldern mit Jahrhundertangabe
<b>_FmtTimeSeconds</b>	Bei Datentyp Zeit werden Sekunden berücksichtigt
<b>_FmtTimeHSeconds</b>	Bei Datentyp Zeit werden Hundertstelsekunden berücksichtigt. Diese Option sollte sinnvollerweise mit <b>_FmtTimeSeconds</b> kombiniert werden.

## Kontakt

Beim Datentyp Gleitkomma kann im Parameter (aPostcomma) die Anzahl der Nachkommastellen übergeben werden. Alle anderen Datentypen kann hier 0 angegeben werden. Der Parameter (aWriteFlag) gibt an, ob die in (aFldBuf) bereitgestellten Daten in den Puffer von (aFldNo) geschrieben werden (true) oder der Inhalt des Feldpuffers (aFldNo) nach (aFldBuf) kopiert wird (false).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte). Im Falle eines negativen Wertes ist ein Fehler aufgetreten und die Daten sind nicht übertragen worden.

## Kontakt

**C16\_FldDataRaw(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vINT aFldNo,  
void\* aFldBuf,  
const vBOOL aWriteFlag) :**  
**vERROR**

Feld binär übertragen

aInstHdl Instanz-Handle

aFileNo Dateinummer

aSbrNo Teildatensatznummer

aFldNo Feldnummer

Zeiger auf

aFldBuf Feldpuffer

Datensatz schreiben

aWriteFlag (true), lesen (false)

Resultat vERROR Fehlerwert

Befehle der

Programmierschnittstelle,

Siehe C16\_ExcFldData(),

C16\_FldData(),

C16\_FldDataChar()

Liest den Inhalt eines alphanumerischen Feldes in einen Puffer bzw. schreibt den Inhalt eines Puffers in ein alphanumerisches Feld. Daten werden binär - also Byte für Byte - ausgelesen bzw. geschrieben, eine Typprüfung findet nicht statt. Dabei wird jeweils die maximale Anzahl von Bytes übertragen (entspricht der definierten Länge des alphanumerischen Feldes). Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion

C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Parameter (aFileNo), (aSbrNo) und (aFldBuf) identifizieren eindeutig das Feld, das ausgelesen bzw. geschrieben werden soll. Der Parameter (aWriteFlag) gibt an, ob die in (aFldBuf) bereitgestellten Daten in den Puffer von (aFldNo) geschrieben werden (true) oder der Inhalt des Feldpuffers (aFldNo) nach (aFldBuf) kopiert wird (false).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

**C16\_RecDelete(const vHANDLE aInstHdl, const vINT aFileNo, const vFLAGS aFlags) : vERROR**  
Datensatz löschen  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aFlags Löschposition  
Resultat vERROR Fehlerwert

Siehe [Befehle der Programmierschnittstelle](#),  
[C16\\_RecDeleteAll\(\)](#),  
[C16\\_RecInsert\(\)](#)

Mit dieser Funktion kann ein Datensatz in der Datei (aFileNo) gelöscht werden. Dabei wird immer über den ersten Schlüssel zugegriffen. Der Satz wird nur bei dem Ergebnis \_rOk gelöscht. Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. Der Parameter (aFlags) bestimmt, welcher Datensatz gelöscht wird. Folgende symbolischen Konstanten können angegeben werden:

- \_RecFirst Der erste Satz wird gelöscht.
- \_RecLast Der letzte Satz wird gelöscht.
- \_RecNext Der nächste Satz wird gelöscht.
- \_RecPrev Der vorherige Satz wird gelöscht.

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- 0 \_rOK** Datensatzoperation erfolgreich.
- 1 \_rLocked** Der Datensatz konnte nicht gelöscht werden, da er von einem anderen Benutzer gesperrt ist.
- 3 \_rNoKey** In der Datei ist kein Satz mit dem gewünschten Schlüsselwert vorhanden. Es wurde der Satz mit dem nächst größeren Schlüsselwert geladen.
- 4 \_rLastRec** In der Datei ist weder ein Satz mit dem gewünschten Schlüsselwert noch ein Satz mit einem größeren Schlüsselwert vorhanden. Es wurde der Satz mit dem größten Schlüsselwert geladen.
- 5 \_rNoRec** Es wurde kein Satz geladen, da entweder die Datei leer ist, oder kein vorhergehender bzw. nachfolgender Satz existiert.
- 10 \_rDeadlock** Der Datensatz konnte aufgrund einer Verklemmung nicht gelöscht werden.

## Kontakt

**C16\_RecDeleteAll(const vHANDLE  
aInstHdl, const vINT aFileNo) :**  
**vERROR**  
Alle Datensätze einer Datei löschen  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
Resultat vERROR Fehlerwert

Siehe  
Befehle der  
Programmierschnittstelle,

C16\_RecDelete(),  
C16\_RecInsert()

Diese Funktion löscht den kompletten Inhalt der Datei (aFileNo). Diese Anweisung kann auch im Mehrbenutzerbetrieb eingesetzt werden. Dabei werden allerdings auch alle eventuell gesperrten Datensätze in der angegebenen Datei gelöscht. Je nach Größe der Datei benötigt C16\_RecDeleteAll() eine gewisse Zeit zum Löschen. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_RecInsert(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vFLAGS aFlags) : vERROR**  
**Datensatz einfügen**  
**aInstHdl Instanz-Handle**  
**aFileNo Dateinummer**  
    **Datensatz sperren**  
**aFlags (\_RecLock), nicht**  
        **sperren (0)**  
**Resultat vERROR Fehlerwert**

**Siehe**  
    Befehle der Programmierschnittstelle,  
    C16\_RecDelete(),  
    C16\_RecReplace()

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Mit dieser Funktion wird der momentan im Speicher stehende Datensatz (bestehend aus den aktuellen Feldinhalten) als neuer Satz in die angegebene Datei (aFileNo) eingefügt. Wird als Parameter (aFlags) die symbolische Konstante \_RecLock übergeben, ist der Datensatz nach dem Einfügen gesperrt.

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- 0 \_rOk**      Datensatzoperation erfolgreich.
- 6 \_rExists**    Der Datensatz konnte nicht eingefügt oder zurückgespeichert werden,  
                  da ein Satz mit einem identischen eindeutigen Schlüsselwert bereits  
                  existiert.
- 10 \_rDeadlock** Der Datensatz konnte aufgrund einer Verklemmung nicht eingefügt  
                  werden.

## Kontakt

**C16\_RecLink(const vHANDLE aInstHdl,**  
**const vINT aSrcFileNo,**  
**const vINT aDstFileNo,**  
**const vXLONG aLinkNo,**  
**const vFLAGS aFlags,**  
**const vINT unused,**  
**const vINT aLinkPos) : vERROR**

Verknüpften Datensatz lesen  
aInstHdl Instanz-Handle  
aSrcFileNo Nummer der Ausgangsdatei  
aDstFileNo Zieldatei  
aLinkNo Nummer der Verknüpfung  
aFlags Zugriffsposition unbenutzt, muss 0  
unused enthalten Zusätzliche  
aLinkPos Information zur Zugriffsposition  
Resultat vERROR Fehlerwert Befehle der  
Siehe Programmierschnittstelle,  
C16\_RecRead()

Der Befehl liest einen verknüpften Datensatz und überträgt ihn in die Feldpuffer.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Funktion führt einen Zugriff auf die Datei (aDstFileNo) durch. Aus dem zuvor gelesenen Datensatz aus der Datei (aScrFileNo) werden aus den Verknüpfungsfeldern (inkl. der Felder für Zugriffspositionierung) der Verknüpfung (aLinkNo) ein Schlüsselwert gebildet und auf den Satz in der Zieldatei zugegriffen. Ist kein Datensatz mit dem entsprechenden Schlüssel vorhanden, wird der Datensatz mit dem nächst größeren bzw. dem größten Schlüssel geladen.

Soll auf eine Selektionsmenge zugegriffen werden, wird in dem Parameter (aLinkNo) der in Ganzzahlig gewandelte Selektionsdeskriptor übergeben. Der Selektionsdeskriptor muss zuvor mit C16\_SelOpen() angelegt worden sein.

\_RecFirst Der erste verknüpfte Satz wird geladen.  
\_RecLast Der letzte verknüpfte Satz wird geladen.  
\_RecPrev Der vorherige verknüpfte Satz wird geladen. Sofern kein weiterer verknüpfter Satz vorhanden ist, wird als Resultat \_rNoRec zurückgeliefert.  
\_RecNext Der nächste verknüpfte Satz wird geladen. Sofern kein weiterer verknüpfter Satz vorhanden ist, wird als Resultat \_rNoRec zurückgeliefert.

## Kontakt

<u>_RecPos</u>	Der Zugriff findet über die Verknüpfungsposition statt (vgl. <a href="#">C16_RecLinkInfo()</a> ), die in (aLinkPos) übergeben werden muss. Da die Position exakt verarbeitet wird, liegen gültige Werte im Bereich von 1 bis zur Anzahl verknüpfter Sätze, andernfalls wird _rNoRec zurückgeliefert.
<u>_RecKeyReverse</u>	Die Schlüsselreihenfolge in der mit _RecPrev, _RecNext und _RecPos zugegriffen wird, kann mit dieser Konstanten umgekehrt werden.
<u>_RecLock</u>	Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn _rOk zurückgeliefert wird. Ist der Satz durch einen anderen Benutzer gesperrt, ist das Resultat _rLocked.
<u>_RecForceLock</u>	Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn _rOk zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, so kann dieser Benutzer den Satz nicht mehr zurückspeichern. Daher sollte _RecForceLock nur in den Fällen erfolgen, in denen ein Satz unbedingt und ohne Rücksicht auf andere Benutzer gesperrt werden muss. Die Sperre schlägt fehl, wenn der Datensatz durch andere Benutzer gemeinsam (_RecSharedLock) gesperrt ist oder der Datensatz in diesem Moment gerade geschrieben oder gelöscht wird. Wird _rLocked zurückgegeben, kann der Befehl nach kurzer Zeit erneut versucht werden.
	 Fahrlässiger Umgang mit dieser Option kann zur Inkonsistenz des Datenbestandes führen.
<u>_RecUnlock</u>	Der gelesene Satz wird entsperrt.
<u>_RecCheckLock</u>	Der Sperrstatus des Satzes wird überprüft. Das Resultat ist _rLocked anstelle von _rOk, wenn ein anderer Benutzer den Satz gesperrt hat. Der Sperrstatus wird nicht verändert.
<u>_RecTest</u>	Hierbei erfolgt der Zugriff nur auf den Schlüssel der Zieldatei. Es wird nicht auf den Datensatz zugegriffen, daher ist die Angabe einer Sperroption ohne Wirkung.
<u>_RecNoLoad</u>	Der gelesene Datensatz wird nicht in die Feldpuffer übertragen. Wird in (aFlag) die Option _RecPos angegeben, muss in (aLinkPos) die Position des zu lesenden Datensatzes übergeben werden. Der Parameter (unused) wird nicht benutzt, er muss 0 enthalten.

Soll der verknüpfte Datensatz aus einer Selektionsmenge gelesen werden, wird anstelle der Verknüpfungsnummer der Deskriptor des Selektionspuffers angegeben. Dabei muss ein Cast des Deskriptors auf vINT verwendet werden.

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

**0 \_rOK** Datensatzoperation erfolgreich.

**1 \_rLocked** Datensatz ist vorhanden und von einem anderen Benutzer gesperrt. Der Satz wurde geladen, sofern die Option \_RecNoLoad nicht angegeben wurde.

**5 \_rNoRec**

## **Kontakt**

**Es wurde kein Satz geladen, da entweder die Datei leer ist, oder kein vorhergehender bzw. nachfolgender Satz existiert.**

## Kontakt

**C16\_RecRead(const vPHANDLE**

**aInstHdl,**

**const vINT aFileNo,**

**const vINT aKeyNo,**

**const vFLAGS aFlags,**

**const vLONG aAddInfo) :**

**vERROR**

**Datensatz lesen**

**aInstHdl Instanz-Handle**

**aFileNo Dateinummer**

**aKeyNo Schlüsselnummer**

**aFlags Zugriffsposition**

**aAddInfo Zusätzliche**

**Information**

**Resultat vERROR Fehlerwert**

**Befehle der**

**Siehe Programmierschnittstelle,**

**C16\_RecLink()**



Die Anweisung ist nur noch aus Kompatibilitätsgründen im Sprachumfang enthalten. Es sollte die Methode **C16\_RecRead64** verwendet werden.

Liest einen Datensatz und überträgt ihn in die Feldpuffer.

Im Parameter (aInstHdl) wird der von **C16\_InitInstance()** bereitgestellte Instanz-Handle, der in der Funktion **C16\_OpenArea()** zum Öffnen der Datenbank verwendet wurde, übergeben. Die Funktion führt einen Zugriff über den Schlüssel (aKeyNo) in die Datei (aFileNo) durch. Der Schlüssel wird aus den aktuellen Feldinhalten der Datei gebildet. Ist kein Datensatz mit dem entsprechenden Schlüssel vorhanden, so wird der Datensatz mit dem nächst größeren bzw. dem größten Schlüssel geladen. Nach dem erfolgreichen Aufruf der Funktion, steht der gelesene Datensatz in den Feldpuffern.

Soll auf eine Selektionsmenge zugegriffen werden, wird in dem Parameter (aKeyNo) der in Ganzzahlig gewandelte Selektionsdeskriptor übergeben. Der Selektionsdeskriptor muss zuvor mit **C16\_SelOpen()** angelegt worden sein.

Die Zugriffsposition wird über den Parameter (aFlag) angegeben. Folgende symbolische Konstanten können angegeben werden:

**0**

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

## Kontakt

<b>_RecFirst</b>	Der Satz mit dem kleinsten Schlüsselwert wird gelesen.
<b>_RecLast</b>	Der Satz mit dem größten Schlüsselwert wird gelesen.
<b>_RecNext</b>	Der Satz mit dem nächst größeren Schlüsselwert wird geladen. Sofern kein weiterer Satz vorhanden ist, wird als Resultat <b>_rNoRec</b> zurückgeliefert.
<b>_RecPrev</b>	Der Satz mit dem nächst kleineren Schlüsselwert wird geladen. Sofern kein weiterer Satz vorhanden ist, wird als Resultat <b>_rNoRec</b> zurückgeliefert.
<b>_RecKeyReverse</b>	Die Schlüsselreihenfolge in der mit <b>_RecPrev</b> , <b>_RecNext</b> und <b>_RecPos</b> zugegriffen wird, kann mit dieser Konstanten umgekehrt werden.
<b>_RecID</b>	Anstatt über einen Schlüssel wird der Datensatz über die <b>Datensatz-ID</b> gelesen. In (aKeyNo) darf kein Schlüssel oder Selektionsdeskriptor übergeben werden. Die Datensatz-ID kann zuvor mit der Anweisung C16 <b>_RecInfo64(..., _RecID)</b> ermittelt. Im Parameter (aAddInfo) muss die zu lesende Datensatz-ID übergeben werden.
<b>_RecPos</b>	Anstatt über den aktuellen Schlüsselwert wird über die Schlüsselposition zugegriffen. Die Position wird in (aAddInfo) übergeben. Diese Position wird aus nur ungefähr berechnet, da eine genaue Positionierung zu lange dauern würde.
<b>_RecUnlock</b>	Der gelesene Satz wird entsperrt.
<b>_RecLock</b>	Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn <b>_rOk</b> zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, so ist das Resultat <b>_rLocked</b> .
<b>_RecSingleLock</b>	Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn <b>_rOk</b> zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch den eigenen oder einen anderen Benutzer gesperrt, ist das Resultat <b>_rLocked</b> . Das Sperren eines Datensatzes erfolgt über die Benutzer-ID. Mit <b>_RecSingleLock</b> wird der gleiche Datensatz für den gleichen Benutzer nur einmal gesperrt. Beim Versuch, denselben Satz ein zweites Mal zu sperren, wird das Resultat <b>_rLocked</b> zurückgeliefert.
<b>_RecSharedLock</b>	Der gelesene Datensatz wird gemeinsam mit anderen Benutzern gesperrt. Dies ist nur dann der Fall, wenn <b>_rOk</b> zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, ist das Resultat <b>_rLocked</b> . Im Gegensatz zur Option <b>_RecLock</b> können von anderen Benutzern weitere Sperren mit dieser Option eingerichtet werden. Ein mit <b>_RecSharedLock</b> gesperrter Datensatz kann von einem anderen Benutzer nicht mit der Option <b>_RecLock</b> zum Schreiben gesperrt werden, bis die letzte Sperre aufgehoben wurde. Der Benutzer, der die gemeinsame Sperre eingerichtet hat, kann den Datensatz nur dann mit <b>_RecLock</b> sperren, wenn in der Zwischenzeit kein anderer Benutzer eine gemeinsame Sperre eingerichtet hat. Mit dieser Option können mehrere Benutzer einen Datensatz vor Veränderung schützen. Der Datensatz kann mit dieser Sperre nicht zurückgeschrieben werden.
<b>_RecForceLock</b>	Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn

## Kontakt

`_rOk` zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, so kann dieser Benutzer den Satz nicht mehr zurückspeichern. Daher sollte `_RecForceLock` nur in den Fällen erfolgen, in denen ein Satz unbedingt und ohne Rücksicht auf andere Benutzer gesperrt werden muss. Die Sperre schlägt fehl, wenn der Datensatz durch andere Benutzer gemeinsam (`_RecSharedLock`) gesperrt ist oder der Datensatz in diesem Moment gerade geschrieben oder gelöscht wird. Wird `_rLocked` zurückgegeben, kann der Befehl nach kurzer Zeit erneut versucht werden.



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

`_RecCheckLock` Der Sperrstatus des Satzes wird überprüft. Das Resultat ist

`_rLocked` anstelle von `_rOk`, wenn ein anderer Benutzer den Satz gesperrt hat. Der Sperrstatus wird nicht verändert.

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

`_RecNoLoad` Der gelesene Datensatz wird nicht in die Feldpuffer übertragen.

Der Parameter (`aAddInfo`) ist optional, er hat zwei verschiedene Bedeutungen. Bei der Benutzung der Option `_RecPos` wird hier die gewünschte Schlüsselposition übergeben. Bei der Option `_RecID` wird die ID des gewünschten Satzes übergeben. Die Optionen `_RecTest` und `_RecPos` sind dabei nicht möglich, da kein Schlüsselzugriff erfolgt. Dadurch gibt es auch die Resultate `_rNoKey` und `_rMultiKey` nicht. Ist der Satz mit der angegebenen ID nicht vorhanden, so wird `_rNoRec` zurückgeliefert und kein Satz geladen.

Soll der Datensatz aus einer Selektionsmenge gelesen werden, wird anstelle der Schlüsselnummer (`aKeyNo`) der Deskriptor des Selektionspuffers angegeben. Dabei muss ein Cast des Deskriptors auf `vINT` verwendet werden.

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- |                     |  |
|---------------------|--|
| <b>0 _rOK</b>       | Datensatzoperation erfolgreich.  |
| <b>1 _rLocked</b>   | Datensatz ist vorhanden und von einem anderen Benutzer gesperrt. Der Satz wurde geladen, sofern die Option <code>_RecNoLoad</code> nicht angegeben wurde.                            |
| <b>2 _rMultiKey</b> | Der Schlüssel ist nicht eindeutig. In der Datei sind mehrere Sätze mit dem gewünschten Schlüsselwert vorhanden, der erste Satz wurde geladen.  |
| <b>3 _rNoKey</b>    | In der Datei ist kein Satz mit dem gewünschten Schlüsselwert vorhanden. Es wurde der Satz mit dem nächst größeren Schlüsselwert geladen.   |
| <b>4 _rLastRec</b>  | In der Datei ist weder ein Satz mit dem gewünschten Schlüsselwert noch ein Satz mit einem größeren Schlüsselwert vorhanden. Es wurde der Satz mit dem größten Schlüsselwert geladen. |
| <b>5 _rNoRec</b>    |  |

## **Kontakt**

**Es wurde kein Satz geladen, da entweder die Datei leer ist, oder kein vorhergehender bzw. nachfolgender Satz existiert.**

## Kontakt

**C16\_RecRead64(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vXLONG aKeyNo,  
const vFLAGS aFlags,  
const vXLONG aAddInfo) :**  
**vERROR**  
Datensatz lesen  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aKeyNo Schlüsselnummer  
aFlags      Zugriffsposition  
              Zusätzliche  
              aAddInfo

### Information

Resultat vERROR Fehlerwert

Befehle der  
Siehe [Programmierschnittstelle,](#)  
[C16\\_RecLink\(\)](#)

Liest einen Datensatz und überträgt ihn in die Feldpuffer.

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. Die Funktion führt einen Zugriff über den Schlüssel (aKeyNo) in die Datei (aFileNo) durch. Der Schlüssel wird aus den aktuellen Feldinhalten der Datei gebildet. Ist kein Datensatz mit dem entsprechenden Schlüssel vorhanden, so wird der Datensatz mit dem nächst größeren bzw. dem größten Schlüssel geladen. Nach dem erfolgreichen Aufruf der Funktion, steht der gelesene Datensatz in den Feldpuffern.

Soll auf eine Selektionsmenge zugegriffen werden, wird in dem Parameter (aKeyNo) der in Ganzzahlig gewandelte Selektionsdeskriptor übergeben. Der Selektionsdeskriptor muss zuvor mit [C16\\_SelOpen\(\)](#) angelegt worden sein.

Die Zugriffsposition wird über den Parameter (aFlag) angegeben. Folgende symbolische Konstanten können angegeben werden:

- |                  |  |
|------------------|--|
| 0                | Ausgehend vom Inhalt der Feldpuffer wird genau der angegebene Datensatz gelesen. Der Rückgabewert des Befehls ist dann abhängig, ob über einen eindeutigen oder mehrdeutigen Schlüssel gelesen wurde _rOk oder _rMultiKey. Eine Satzsperrre beeinflusst den Rückgabewert nicht. Konnte aus dem Inhalt der Schlüsselfelder kein vorhandener Schlüsselwert gebildet werden, wird der darauf folgende Datensatz gelesen. In diesem Fall gibt der Befehl den Wert _rNoKey zurück. Gibt es keinen nachfolgenden Datensatz, wird der letzte Datensatz gelesen. Zurückgegeben wird dann der Wert _rLastRec. Konnte kein Datensatz gelesen werden (d. h. die Datei ist leer), wird der Wert _rNoRec zurückgegeben. |
| <u>_RecFirst</u> | Der Satz mit dem kleinsten Schlüsselwert wird gelesen.   |
| <u>_RecLast</u>  | Der Satz mit dem größten Schlüsselwert wird gelesen.   |

## Kontakt

<u>_RecNext</u>	<p><b>Der Satz mit dem nächst größeren Schlüsselwert wird geladen.</b> Sofern kein weiterer Satz vorhanden ist, wird als Resultat <u>_rNoRec</u> zurückgeliefert.</p>
<u>_RecPrev</u>	<p><b>Der Satz mit dem nächst kleineren Schlüsselwert wird geladen.</b> Sofern kein weiterer Satz vorhanden ist, wird als Resultat <u>_rNoRec</u> zurückgeliefert.</p>
<u>_RecKeyReverse</u>	<p><b>Die Schlüsselreihenfolge in der mit <u>_RecPrev</u>, <u>_RecNext</u> und <u>_RecPos</u> zugegriffen wird, kann mit dieser Konstanten umgekehrt werden.</b></p>
<u>_RecID</u>	<p>Anstatt über einen Schlüssel wird der Datensatz über die <b>Datensatz-ID</b> gelesen. In (aKeyNo) darf kein Schlüssel oder Selektionsdeskriptor übergeben werden. Die Datensatz-ID kann zuvor mit der Anweisung C16 <u>RecInfo64(..., _RecID)</u> ermittelt. Im Parameter (aAddInfo) muss die zu lesende Datensatz-ID übergeben werden.</p>
<u>_RecPos</u>	<p>Anstatt über den aktuellen Schlüsselwert wird über die Schlüsselposition zugegriffen. Die Position wird in (aAddInfo) übergeben. Diese Position wird aus nur ungefähr berechnet, da eine genaue Positionierung zu lange dauern würde.</p>
<u>_RecUnlock</u>	<p>Der gelesene Satz wird entsperrt.</p>
<u>_RecLock</u>	<p>Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn <u>_rOk</u> zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, so ist das Resultat <u>_rLocked</u>.</p>
<u>_RecSingleLock</u>	<p>Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn <u>_rOk</u> zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch den eigenen oder einen anderen Benutzer gesperrt, ist das Resultat <u>_rLocked</u>. Das Sperren eines Datensatzes erfolgt über die Benutzer-ID. Mit <u>_RecSingleLock</u> wird der gleiche Datensatz für den gleichen Benutzer nur einmal gesperrt. Beim Versuch, denselben Satz ein zweites Mal zu sperren, wird das Resultat <u>_rLocked</u> zurückgeliefert.</p>
<u>_RecSharedLock</u>	<p>Der gelesene Datensatz wird gemeinsam mit anderen Benutzern gesperrt. Dies ist nur dann der Fall, wenn <u>_rOk</u> zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, ist das Resultat <u>_rLocked</u>. Im Gegensatz zur Option <u>_RecLock</u> können von anderen Benutzern weitere Sperren mit dieser Option eingerichtet werden. Ein mit <u>_RecSharedLock</u> gesperrter Datensatz kann von einem anderen Benutzer nicht mit der Option <u>_RecLock</u> zum Schreiben gesperrt werden, bis die letzte Sperre aufgehoben wurde. Der Benutzer, der die gemeinsame Sperre eingerichtet hat, kann den Datensatz nur dann mit <u>_RecLock</u> sperren, wenn in der Zwischenzeit kein anderer Benutzer eine gemeinsame Sperre eingerichtet hat. Mit dieser Option können mehrere Benutzer einen Datensatz vor Veränderung schützen. Der Datensatz kann mit dieser Sperre nicht zurückgeschrieben werden.</p>
<u>_RecForceLock</u>	<p>Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn <u>_rOk</u> zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, so kann dieser Benutzer den Satz nicht mehr zurückspeichern. Daher sollte <u>_RecForceLock</u> nur in den Fällen</p>

## Kontakt

erfolgen, in denen ein Satz unbedingt und ohne Rücksicht auf andere Benutzer gesperrt werden muss. Die Sperre schlägt fehl, wenn der Datensatz durch andere Benutzer gemeinsam (\_RecSharedLock) gesperrt ist oder der Datensatz in diesem Moment gerade geschrieben oder gelöscht wird. Wird \_rLocked zurückgegeben, kann der Befehl nach kurzer Zeit erneut versucht werden.



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

**\_RecCheckLock** Der Sperrstatus des Satzes wird überprüft. Das Resultat ist

\_rLocked anstelle von \_rOk, wenn ein anderer Benutzer den Satz gesperrt hat. Der Sperrstatus wird nicht verändert.

**\_RecTest**

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

**\_RecNoLoad**

Der gelesene Datensatz wird nicht in die Feldpuffer übertragen.

Der Parameter (aAddInfo) ist optional, er hat zwei verschiedene Bedeutungen. Bei der Benutzung der Option \_RecPos wird hier die gewünschte Schlüsselposition übergeben. Bei der Option \_RecID wird die ID des gewünschten Satzes übergeben. Die Optionen \_RecTest und \_RecPos sind dabei nicht möglich, da kein Schlüsselzugriff erfolgt. Dadurch gibt es auch die Resultate \_rNoKey und \_rMultiKey nicht. Ist der Satz mit der angegebenen ID nicht vorhanden, so wird \_rNoRec zurückgeliefert und kein Satz geladen.

Soll der Datensatz aus einer Selektionsmenge gelesen werden, wird anstelle der Schlüsselnummer (aKeyNo) der Deskriptor des Selektionspuffers angegeben. Dabei muss ein Cast des Deskriptors auf vINT verwendet werden.

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- |                     |  |
|---------------------|--|
| <b>0 _rOK</b>       | Datensatzoperation erfolgreich.  |
| <b>1 _rLocked</b>   | Datensatz ist vorhanden und von einem anderen Benutzer gesperrt. Der Satz wurde geladen, sofern die Option _RecNoLoad nicht angegeben wurde.   |
| <b>2 _rMultiKey</b> | Der Schlüssel ist nicht eindeutig. In der Datei sind mehrere Sätze mit dem gewünschten Schlüsselwert vorhanden, der erste Satz wurde geladen.  |
| <b>3 _rNoKey</b>    | In der Datei ist kein Satz mit dem gewünschten Schlüsselwert vorhanden. Es wurde der Satz mit dem nächst größeren Schlüsselwert geladen.   |
| <b>4 _rLastRec</b>  | In der Datei ist weder ein Satz mit dem gewünschten Schlüsselwert noch ein Satz mit einem größeren Schlüsselwert vorhanden. Es wurde der Satz mit dem größten Schlüsselwert geladen. |
| <b>5 _rNoRec</b>    | Es wurde kein Satz geladen, da entweder die Datei leer ist, oder kein vorhergehender bzw. nachfolgender Satz existiert.  |

## Kontakt

**C16\_RecReplace(const vHANDLE aInstHdl, const vINT aFileNo, const vFLAGS aFlags) : vERROR**  
Datensatz ersetzen  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aFlags  
    Datensatz sperren  
    beibehalten (0),  
    Datensatz entsperren

Resultat vERROR Fehlerwert

Siehe Befehle der Programmierschnittstelle,  
C16\_RecDelete(),  
C16\_RecInsert()

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Diese Funktion speichert einen Datensatz in die angegebene Datei (aFileNo) zurück, der zuvor geladen und gesperrt wurde. Ein nicht gesperrter Satz kann nicht zurückgespeichert werden. Alle Felder des gelesenen Satzes können vor dem Rückspeichern verändert werden (inkl. aller Schlüsselfelder). Dabei ist zu beachten, dass eindeutige Schlüsselwerte nicht bereits in der Datei enthalten sein dürfen. Durch (aFlags) kann angegeben werden, ob der Datensatz nach dem Rückspeichern weiter gesperrt bleiben soll oder nicht. Wird keine der beiden Optionen benutzt, so wird der Satz entsperrt.

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- 0 \_rOk Datensatzoperation erfolgreich.
- 6 \_rExists Der Datensatz konnte nicht zurückgespeichert werden, da ein Satz mit einem identischen eindeutigen Schlüsselwert bereits existiert.
- 7 \_rNoLock Der Datensatz konnte nicht zurückgespeichert werden, da er nicht gesperrt ist.
- 10 \_rDeadlock Der Datensatz konnte aufgrund einer Verklemmung nicht ersetzt werden.

## Kontakt

**C16\_SbrSetStatus(const vPHANDLE  
aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vBOOL aNewStatus) : vERROR**  
Teildatensatz aktivieren / deaktivieren  
aInstHdl      Instanz-Handle  
aFileNo      Dateinummer  
aSbrNo      Teildatensatznummer  
aNwStatus      Teildatensatz aktivieren  
                  (true), deaktivieren (false)  
**Resultat      vERROR      Fehlerwert**  
**Siehe          Programmierschnittstelle**

Der Status des bedingten Teildatensatzes (nicht feldabhängig) mit der Nummer (aSbrNo) in der Datei (aFileNo) wird aktiviert (aNwStatus = true) bzw. nicht aktiviert (aNwStatus = false).

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_SetExcMode(const vHANDLE aInstHdl, const  
vINT aFileNo,  
const vBOOL aAutoFlag) : vERROR**  
Modus des automatischen Variablenabgleichs festlegen aInstHdl

Instanz-Handle

aFileNo Dateinummer

Automatischen Transfer

aAutoFlag aktivieren (true),  
deaktivieren (false)

Resultat vERROR

Fehlerwert

Befehle der

Siehe Programmierschnittstelle,  
C16\_RegExtFld()

Diese Funktion sorgt vor und nach den Datensatz-Operationen C16\_RecRead, C16\_RecLink(), C16\_RecInsert(), C16\_RecReplace() und C16\_RecDelete() für den automatischen Transfer von Daten zwischen allen Feldern im Feldpuffer der Datei und den mit ihnen verbundenen Programmvariablen (vgl. C16\_RegExtFld()), damit diese den aktuellen Inhalt des Feldpuffers widerspiegeln. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion

C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Der automatische Transfer wird mit (aAutoFlag) = true aktiviert und findet solange statt, bis der Modus durch einen erneuten Aufruf mit (aAutoFlag) = false wieder deaktiviert wird.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

Externe Windows Programmierschnittstelle - Satzpuffer-Funktionen Übersicht über die Satzpuffer-Funktionen der Programmierschnittstelle

- [C16\\_RecBufClear](#)
- [C16\\_RecBufCompare](#)
- [C16\\_RecBufCopy](#)
- [C16\\_RecBufCreate](#)
- [C16\\_RecBufDestroy](#)
- [C16\\_RegExtFld](#)
- [C16\\_SbrClear](#)

## Kontakt

```
C16_RecBufCreate(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
vHANDLE* aBuffHdl) :  
vERROR  
Datensatzpuffer anlegen  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
          Zeiger auf  
aBuffHdl  
          Datensatzpuffer
```

### Befehle der

Siehe [Programmierschnittstelle,](#)  
[C16\\_RecBufDestroy\(\)](#)

Mit dieser Funktion wird ein zusätzlicher Datensatzpuffer für die Datei (aFileNo) im Hauptspeicher angelegt. Mit dem Handle des Puffers können anschließend die Funktionen [C16\\_RecBufCopy\(\)](#) und [C16\\_RecBufCompare\(\)](#) aufgerufen werden. Wenn der Puffer nicht mehr benötigt wird, muss er mit [C16\\_RecBufDestroy\(\)](#) entfernt werden. Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. In (aFileNo) steht die Dateinummer der Datei, für die ein zusätzlicher Datensatzpuffer angelegt werden soll. (aBuffHdl) ist ein Zeiger auf eine Variable vom Typ vHANDLE. Bei erfolgreicher Ausführung wird der Puffer-Handle in diese Variable geschrieben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_RecBufClear(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vHANDLE aBuffHdl) :**  
**vERROR**  
Datensatzpuffer leeren  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aBuffHdl Puffer-Handle  
Resultat vERROR Fehlerwert

Siehe [Befehle der Programmierschnittstelle](#),  
[C16\\_RecBufCopy\(\)](#),  
[C16\\_SbrClear\(\)](#)

Leert alle Felder im Datensatzpuffer der Datei bzw. in einem mit [C16\\_RecBufCreate\(\)](#) erzeugten Datensatzpuffer. Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. In (aFileNo) oder (aBuffHdl) wird die Datei angegeben, deren Puffer geleert werden soll. Wurde der Puffer zuvor mit [C16\\_RecBufCreate\(\)](#) erzeugt, ist der Puffer-Handle anzugeben, sonst steht in (aBuffHdl) NULL.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_SbrClear(const vHANDLE  
aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo) : vERROR**  
Teildatensatzpuffer leeren  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aSbrNo Teildatensatznummer  
Resultat vERROR Fehlerwert

Befehle der

Siehe Programmierschnittstelle,  
C16\_RecBufClear()

Leert alle Felder des Teildatensatzes mit Nummer (aSbrNo) in der Datei mit der Nummer (aFileNo). Dabei werden keine Vorgabenwerte oder Zähler eingetragen. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Parameter (aFileNo) und (aSbrNo) identifizieren den Teildatensatz, deren Feldpuffer geleert werden sollen.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_RecBufCopy(const  
vHANDLE aInstHdl,  
const vINT aSrcFileNo,  
const vINT aDstFileNo,  
const vHANDLE aSrcBuffHdl,  
const vHANDLE aDstBuffHdl) :**  
**vERROR**  
**Datensatzpuffer kopieren**  
**aInstHdl** Instanz-Handle  
**aSrcFileNo** Nummer der  
Quelldatei  
**aDstFileNo** Nummer der Zieldatei der  
Handle des  
aSrcBuffHdl Quell-Puffers  
Handle des  
Ziel-Puffers  
**Resultat** vERROR Fehlerwert  
Befehle der  
**Siehe** Programmierschnittstelle,  
C16\_RecBufCompare()

Diese Funktion kopiert den aktuellen Inhalt des Datensatzpuffers aus der Quelldatei (aSrcFileNo) in den Puffer der Zieldatei (aDstFileNo). In diesem Fall wird in den Parametern (aSrcBuffHdl) und (aDstBuffHdl) NULL übergeben. Wird anstelle einer Dateinummer der Handle eines mit C16\_RecBufCreate() erzeugten Datensatzpuffers angegeben, so muss die Struktur der beiden Puffer übereinstimmen. In diesem Fall wird auch der aktuelle Satz-ID kopiert. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Mit dieser Funktion können auch Dateiinhalte unterschiedlicher Struktur kopiert werden. In diesem Fall wird, beginnend mit dem ersten Teildatensatz, solange Feld für Feld kopiert bis unterschiedliche Feldtypen zwischen Quell- und Zielpuffer festgestellt werden.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_RecBufCompare(const vHANDLE aInstHdl,  
const vINT aSrcFileNo,  
const vINT aDstFileNo,  
const vHANDLE aSrcBuffHdl,  
const vHANDLE aDstBuffHdl) :**  
**vERROR**  
Datensatzpuffer vergleichen  
aInstHdl Instanz-Handle  
aSrcFileNo Nummer der  
aSrcBuffHdl Quelldatei  
aDstFileNo Nummer Zieldatei der  
aDstBuffHdl Handle des  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_RecBufCopy()

Diese Funktion vergleicht den aktuellen Inhalt des Datensatzpuffers aus Datei (aSrcFileNo) mit dem Puffer der Datei (aDstFileNo). Wird anstelle einer Dateinummer der Handle eines mit C16\_RecBufCreate() erzeugten Datensatzpuffers angegeben, so muss die Struktur der beiden Puffer übereinstimmen. Wird der Vergleich anhand der Dateinummern durchgeführt, wird in den Parametern (aScrBuffHdl) und (aDstBuffHdl) NULL übergeben. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (Datensatzpuffer stimmen überein), 1 (Datensatzpuffer weichen voneinander ab) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_RecBufDestroy**(const  
vHANDLE aInstHdl,  
vHANDLE aBuffHdl) : vERROR  
Datensatzpuffer entfernen  
aInstHdl Instanz-Handle  
aBuffHdl Puffer-Handle  
Resultat vERROR Fehlerwert

Befehle der  
Siehe Programmierschnittstelle,  
C16\_RecBufCreate()

Diese Funktion entfernt einen mit C16\_RecBufCreate() erzeugten Datensatzpuffer aus dem  
Hauptspeicher. Der Handle kann anschließend nicht mehr verwendet werden. Im Parameter  
(aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion  
C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen  
Fehlerwert (siehe Fehlerwerte).

## Kontakt

```
C16_RegExtFld(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vINT aFldNo,  
const vFLAGS aExcFlags,  
const vFLAGS aFormatFlags,  
const vINT aPostcomma,  
void* aExtBuffer) : vERROR
```

Feld registrieren

aInstHdl	Instanz-Handle
aFileNo	Dateinummer
aSbrNo	Teildatensatznummer
aFldNo	Feldnummer
aExcFlags	Übertragungsart
aFormatFlags	Formatierung
aPostcomma	Nachkommastellen Zeiger auf
aExtBuffer	Feldpuffer
Resultat	vERROR Fehlerwert <u>Befehle der</u> <u>Programmierschnittstelle,</u> <u>C16_ExcFileData()</u> ,

Siehe

C16\_ExcSbrData(),  
C16\_ExcFldData(),  
C16\_SetExcMode()

Der Befehl erstellt eine direkte Verbindung zwischen einer im Programm definierten Variable und einem Feld im Feldpuffer der Datenstruktur und registriert sie. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Parameter (aFileNo), (aSbrNo) und (aFldBuf) identifizieren eindeutig das Feld, welches registriert werden soll. Der Parameter (aExcFlags) legt fest, wie die Daten übertragen werden. Folgende symbolische Konstanten stehen zur Verfügung:

_FldExc_Binary	Der Daten werden binär - also Byte für Byte - zwischen Feldpuffer und externem Feld übertragen.
_FldExc_Character	Beim Transfer wird der Feldinhalt in eine ASCII-Zeichenkette entsprechend den mit <u>C16_SetCharDefs()</u> festgelegten Einstellungen konvertiert.
_FldExc_RawData	Diese Option ist nur bei alphanumerischen Feldern wirksam (vgl. <u>C16_FldDataRaw()</u> ). Daten werden binär - also Byte für Byte - ausgelesen bzw. geschrieben, eine Typprüfung findet nicht statt. Dabei wird jeweils die maximale Anzahl von Bytes übertragen (entspricht der definierten Länge des alphanumerischen Felds).

Die Art der Formatierung wird in den Parametern (aFormatFlags) und (aPostkomma) angegeben. In (aFormatFlags) kann eine oder mehrere (logische OR-Verknüpfung) der nachfolgenden symbolischen Konstanten angegeben werden:

_FmtNone	keine bestimmte Formatierung
----------	------------------------------

## Kontakt

<b>_FmtTSep</b>	<b>Tausendertrennung</b>
<b>_FmtZeroSupr</b>	<b>Nullunterdruckung</b>
<b>_FmtPlusSign</b>	<b>Positive Werte werden mit einem vorangestellten Plussymbol versehen</b>
<b>_FmtDateLongYear</b>	<b>Jahr bei Datumsfeldern mit Jahrhundertangabe</b>
<b>_FmtTimeSeconds</b>	<b>Bei Datentyp Zeit werden Sekunden berücksichtigt</b>
<b>_FmtTimeHSeconds</b>	<b>Bei Datentyp Zeit werden Hundertstelsekunden berücksichtigt. Diese Option sollte sinnvollerweise mit _FmtTimeSeconds kombiniert werden.</b>

**Bei dem Gleitkomma kann im Parameter (aPostkomma) die Anzahl der Nachkommastellen übergeben werden. Bei allen anderen Datentypen kann hier 0 angegeben werden. Die Formatierung wird verwendet, wenn \_FldExc\_Character angegeben wurde.**

**In (aExtBuffer) wird ein Zeiger auf den Feldpuffer übergeben. Die Funktion kann auch dazu benutzt werden, um eine bereits bestehende Verbindung zu löschen und deren Registrierung aufzuheben, dazu wird in (aExtBuffer) NULL übergeben.**

**Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).**



**Die Registrierung aller Felder im Feldpuffer der Datenstruktur ist nicht immer erforderlich. Es genügt, nur diejenigen Felder zu registrieren, die in der Anwendung tatsächlich bearbeitet werden.**

## Kontakt

Externe Windows Programmierschnittstelle - Funktionen zum Ermitteln von Informationen

Übersicht über die Funktionen zum Ermitteln von Informationen

- [C16 FileInfo](#)
- [C16 FileInfoByName](#)
- [C16 FldInfo](#)
- [C16 FldInfoByName](#)
- [C16 FldInfoExt](#)
- [C16 FldInfoExtByName](#)
- [C16 KeyFldInfo](#)
- [C16 KeyInfo](#)
- [C16 KeyInfoByName](#)
- [C16 LinkFldInfo](#)
- [C16 LinkInfo](#)
- [C16 LinkInfoByName](#)
- [C16 QueryAreaInfo](#)
- [C16 QueryPgifInfo](#)
- [C16 QueryServerInfo](#)
- [C16 QueryUserInfo](#)
- [C16 RecInfo](#)
- [C16 RecInfo64](#)
- [C16 RecLinkInfo](#)
- [C16 SbrInfo](#)
- [C16 SbrInfoByName](#)

## Kontakt

**C16\_QueryPgifInfo(const vHANDLE aPgifHdl,  
vC16\_PgifInfo\* aInfoBlock) : vERROR**

Informationen über die Programmierschnittstelle ermitteln aPgifHdl

Modul-Handle

aInfoBlock Informationsstruktur

Resultat      vERROR      Fehlerwert  
Siehe            Befehle der

Programmierschnittstelle

Der Befehl füllt eine Struktur vom Typ vC16\_PgifInfo mit Informationen zur installierten CONZEPT 16-Programmierschnittstelle. Die Struktur ist folgendermaßen definiert:

```
typedef struct{  vINT  InfoSize;  vINT  PgifType;  vCHAR PgifLicense[20];  vCHAR PgifRelease[8];
```

Komponente	Beschreibung	
InfoSize	Größe der Struktur	
PgifType	Typ des Moduls	
	0x0001 _PgifType_Client	Non-Client-Server
	0x0080 _PgifType_Protocol_16	Windows Client/Server (16-Bit Protokoll)
	0x0100 _PgifType_Win32	Windows Client/Server (32-Bit Protokoll)
	0x0200 _PgifType_OS2	OS/2 32-Bit
PgifLicense[20]	Lizenznummer	
PgifRelease[8]	Release-Nummer	
PgifUserLimit	Anzahl zugelassener Benutzer	
PgifMemory	Aktuell belegter Speicher in Byte	
PgifMemoryPeak	Maximal belegter Speicher seit der Modulinitialisierung in Byte Im Parameter (aPgifHdl) wird der Modul-Handle übergeben, der von <u>C16_InitPgif()</u> bereitgestellt wurde. Die Informationen werden in die Informationsstruktur übertragen, deren Adresse in (aInfoBlock) übergeben wurde. Vor dem Aufruf der Funktion muss die Strukturkomponente InfoSize mit der Größe der Struktur initialisiert werden (sizeof(vC16_PgifInfo)).	

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_QueryServerInfo(const vHANDLE  
aInstHdl, vC16\_ServerInfo\* aInfoBlock) :**  
**vERROR**

Informationen über den Server ermitteln

aInstHdl Instanz-Handle

aInfoBlock Informationsstruktur

Resultat      vERROR      Fehlerwert  
                Befehle der

Siehe

Programmierschnittstelle

Der Befehl füllt eine Struktur vom Typ vC16\_ServerInfo mit Informationen zum Server. Die Struktur ist folgendermaßen definiert:

```
typedef struct{  vINT  InfoSize;  vCHAR ServerName[64];  vCHAR ServerType[48];  vCHAR ServerLicen
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
ServerName[64]	Name des Servers
ServerType[48]	Betriebssystem des Servers
ServerLicense[20]	Lizenznummer
ServerRelease[8]	Release-Nummer
ServerProtocol	Integerwert, der das Protokoll des Servers angibt: 2 _ProtocolTCP TCP/IP
ServerUserLimit	Anzahl zugelassener Benutzer

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Informationen werden in die Informationsstruktur übertragen, deren Adresse in (aInfoBlock) übergeben wurde. Vor dem Aufruf der Funktion muss die Strukturkomponente InfoSize mit der Größe der Struktur initialisiert werden (sizeof(vC16\_ServerInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_QueryAreaInfo(const vPHANDLE  
aInstHdl, vC16\_ServerInfo\* aInfoBlock) :**  
**vERROR**

Informationen über eine Datenbank ermitteln

aInstHdl Instanz-Handle

aInfoBlock Informationsstruktur

Resultat    vERROR    Fehlerwert  
Befehle der

Siehe

Programmierschnittstelle

Der Befehl füllt eine Struktur vom Typ vC16\_AreaInfo mit Informationen zu einer Datenbank. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT UserID; vINT UserNumber; vINT AreaID; vLONG AreaSize
```

Komponente	Beschreibung	
InfoSize	Größe der Struktur	
UserID	Benutzer-ID des aktuellen Benutzers	
UserNumber	Nummer des aktuellen Benutzers	
AreaID	Release-Nummer der CONZEPT 16-Version	
AreaSize	Gesamtgröße des Datenraums in KB	
AreaFree	Größe des freien Datenraums in KB	
AreaUserCount	Anzahl der angemeldeten Benutzer	
AreaStatusFlags	Status der Datenbank	
	0x00000200	_AreaOpen      Datenbank geöffnet
	0x00010000	_AreaStandby      Datenbank im Standby-Modus
	0x00020000	_AreaReadOnly      Datenbank ist schreibgeschützt (Read-Only-Modus)
	0x00040000	_AreaBackup      Datenbank ist im Backup-Modus
	0x00080000	_AreaDiag      Diagnose läuft
	0x00200000	_AreaUpdate      Datenbank ist im Update-Modus
	0x00400000	_AreaLock      Datenbank ist gesperrt
	0x01000000	_AreaOptimize      Optimierung läuft



Es sind nur die Statusflags angegeben, die auch vom Programmierer ausgewertet werden können.

**AreaRemBackupTime** Noch verbleibende Dauer des Sicherungsergebnisses in Sekunden (vgl. [C16\\_ServerBackup\(\)](#)).

**AreaName[128]** Dateiname der Datenbank

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. Die Informationen werden in die Informationsstruktur übertragen, deren Adresse in (aInfoBlock) übergeben wurde. Vor dem Aufruf der Funktion muss die Strukturkomponente InfoSize mit der Größe der Struktur initialisiert werden (sizeof(vC16\_AreaInfo)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_FileInfo**(const vHANDLE  
aInstHdl,  
const vINT aFileNo,  
vC16\_FileInfo\* aInfoBlock) : vERROR  
Informationen zu einer Datei ermitteln  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_FileInfoByName()

Der Befehl füllt eine Struktur vom Typ vC16\_FileInfo mit Informationen zu einer in der Datenstruktur definierten Datei. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT FileNumber; vINT FileMaster; vINT FileSbrCount; vINT
```

### Komponente Beschreibung

InfoSize Größe der Struktur  
FileNumber Dateinummer  
FileMaster Nummer der Hauptdatei (bei einer untergeordneten Datei)  
FileSbrCount Anzahl der Teildatensätze  
FileKeyCount Anzahl der Schlüssel  
FileLinkCount Anzahl der Verknüpfungen  
FileName[24] Dateiname

Als Parameter der Funktion müssen der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde (aInstHdl), die Nummer der Datei, zu der die Informationen ermittelt werden sollen und ein Zeiger auf eine vC16\_FileInfo-Struktur, die mit Informationen zu der Datei gefüllt wird, übergeben werden. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_FileInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_FileInfoByName(const vHANDLE  
aInstHdl,  
const vCHAR\* aFileName,  
vC16\_FileInfo\* aInfoBlock) : vERROR**  
Informationen zu einer Datei ermitteln  
aInstHdl Instanz-Handle  
aFileName Dateiname  
aInfoBlock Informationsstruktur

Resultat vERROR

Fehlerwert

Befehle der

Siehe Programmierschnittstelle,  
C16\_FileInfo()

Der Befehl füllt eine Struktur vom Typ vC16\_FileInfo mit Informationen zu einer in der Datenstruktur definierten Datei. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT FileNumber; vINT FileMaster; vINT FileSbrCount; vINT
```

### Komponente Beschreibung

InfoSize Größe der Struktur

FileNumber Dateinummer

FileMaster Nummer der Hauptdatei (bei einer untergeordneten Datei)

FileSbrCount Anzahl der Teildatensätze

FileKeyCount Anzahl der Schlüssel

FileLinkCount Anzahl der Verknüpfungen

FileName[24] Dateiname

Als Parameter der Funktion müssen der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde (aInstHdl), der Name der Datei, zu der die Informationen ermittelt werden sollen und ein Zeiger auf eine vC16\_FileInfo-Struktur, die mit Informationen zu der Datei gefüllt wird, übergeben werden. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_FileInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_SbrInfo(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
vC16\_SbrInfo\* aInfoBlock) : vERROR Informationen  
zu einem Teildatensatz ermitteln aInstHdl Instanz-  
Handle  
aFileNo Dateinummer  
aSbrNo Teildatensatznummer  
aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_SbrInfoByName()**

Der Befehl füllt eine Struktur vom Typ vC16\_SbrInfo mit Informationen zu einem in der Datenstruktur definierten Teildatensatz. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT SbrFileName; vINT SbrNumber; vBOOL SbrStatus; vINT
```

<b>Komponente</b>	<b>Beschreibung</b>
InfoSize	Größe der Struktur
SbrFileName	Nummer der Datei, in welcher der Teildatensatz definiert ist SbrNumber
Teildatensatznummer	
SbrStatus	Status des Teildatensatzes (0 = Teildatensatz ist nicht aktiv, 1 = Teildatensatz ist aktiv)
SbrFldCount	Anzahl der Felder im Teildatensatz
SbrName[24]	Name des Teildatensatzes

Als Parameter der Funktion müssen der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank übergeben wurde (aInstHdl), die Nummer der Datei und des Teildatensatzes, zu dem die Informationen ermittelt werden sollen und ein Zeiger auf eine vC16\_SbrInfo-Struktur, die mit Informationen zu der Datei gefüllt wird, übergeben werden. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_SbrInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_SbrInfoByName(const vHANDLE aInstHdl,  
const vCHAR aSbrName,  
vC16\_SbrInfo\* aInfoBlock) : vERROR Informationen  
zu einem Teildatensatz ermitteln aInstHdl Instanz-  
Handle  
aSbrName Name des Teildatensatzes  
aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_SbrInfo()**

Der Befehl füllt eine Struktur vom Typ vC16\_SbrInfo mit Informationen zu einem in der Datenstruktur definierten Teildatensatz. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT SbrFileName; vINT SbrNumber; vBOOL SbrStatus; vINT
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
SbrFileName	Nummer der Datei, in welcher der Teildatensatz definiert ist SbrNumber Teildatensatznummer
SbrStatus	Status des Teildatensatzes (0 = Teildatensatz ist nicht aktiv, 1 = Teildatensatz ist aktiv)
SbrFldCount	Anzahl der Felder im Teildatensatz
SbrName[24]	Name des Teildatensatzes

Als Parameter der Funktion müssen der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank übergeben wurde (aInstHdl), der Name des Teildatensatzes, zu dem die Informationen ermittelt werden sollen und ein Zeiger auf eine vC16\_SbrInfo-Struktur, die mit Informationen zu der Datei gefüllt wird, übergeben werden. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_SbrInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_FldInfo(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aSbrNo,  
const vINT aFldNo,  
vC16\_FldInfo\* aInfoBlock) : vERROR**  
Informationen über ein Feld ermitteln  
**aInstHdl** Instanz-Handle  
**aFileNo** Dateinummer  
**aSbrNo** Teildatensatznummer  
**aFldNo** Feldnummer  
**aInfoBlock** Informationsstruktur  
**Resultat** vERROR Fehlerwert  
**Befehle der Programmierschnittstelle,**  
**Siehe** [C16\\_FldInfoByName\(\)](#),  
[C16\\_FldInfoExt\(\)](#),  
[C16\\_FldInfoExtByName\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_FldInfo mit Informationen zu einem in der Datenstruktur definierten Feld. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT FldFileName; vINT FldSbrNumber; vINT FldNumber; vB
```

Komponente	Beschreibung		
InfoSize	Größe der Struktur		
FldFileName	Nummer der Datei, in der das Feld definiert ist		
FldSbrNumber	Nummer des Teildatensatzes, in dem das Feld definiert ist		
FldNumber	Feldnummer		
FldTyp	Type	entsprechender C-Typ	
	1 _TypeAlpha	alphanumerisch	vBYTE[FldLen + 2]
	2 _TypeDate	Datum	vDATE
	4 _TypeWord	ganzzahlig kurz	vWORD
	7 _TypeInt	ganzzahlig lang	vLONGs
	9 _TypeFloat	numerisch	vFLOAT
	10 _TypeLogic	logisch	vBOOL8
	11 _TypeTime	Zeit	vTIME
FldLen	Feldlänge (maximale Länge bei alphanumerischen Feldern)		
FldInputRight	Eingabeberechtigung		
FldOutputRight	Ausgabeberechtigung		

FldName[24] Feldname

Der Parameter (aInstHdl) ist der Instanz-Handle, der von der Funktion

[C16\\_InitInstance\(\)](#) bereitgestellt und an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank übergeben wurde. Zur Identifikation des Feldes werden in den Parametern (aFileNo), (aSbrNo) und (aFldNo) die Datei-, die Teildatensatz- und die Feldnummer

## Kontakt

angegeben. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_FldInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_FldInfo)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_FldInfoByName**(const  
vHANDLE aInstHdl,  
const vCHAR\* aFldName,  
vC16\_FldInfo\* aInfoBlock) : vERROR

Informationen über ein Feld ermitteln  
aInstHdl Instanz-Handle  
aFldName Feldname

aInfoBlock Informationsstruktur

Resultat vERROR Fehlerwert

Befehle der  
Programmierschnittstelle,  
Siehe [C16\\_FldInfo\(\)](#),  
[C16\\_FldInfoExt\(\)](#),  
[C16\\_FldInfoExtByName\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_FldInfo mit Informationen zu einem in der Datenstruktur definierten Feld. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT FldFileNumber; vINT FldSbrNumber; vINT FldNumber; vB
```

**Komponente** Beschreibung

**InfoSize** Größe der Struktur

**FldFileNumber** Nummer der Datei, in der das Feld definiert ist

**FldSbrNumber** Nummer des Teildatensatzes, in dem das Feld definiert ist

**FldNumber** Feldnummer

**FldTyp** Typ entsprechender

C-Typ

1	_TypeAlpha	alphanumerisch	vBYTE[FldLen + 2]
---	------------	----------------	----------------------

2	_TypeDate	Datum	vDATE
---	-----------	-------	-------

4	_TypeWord	ganzzahlig kurz	vWORD
---	-----------	-----------------	-------

7	_TypeInt	ganzzahlig lang	vLONGs
---	----------	-----------------	--------

9	_TypeFloat	numerisch	vFLOAT
---	------------	-----------	--------

10	_TypeLogic	logisch	vBOOL8
----	------------	---------	--------

11	_TypeTime	Zeit	vTIME
----	-----------	------	-------

**FldLen** Feldlänge (maximale Länge bei alphanumerischen Feldern)

**FldInputRight** Eingabeberechtigung

**FldOutputRight** Ausgabeberechtigung

**FldName[24]** Feldname

Der Parameter (aInstHdl) ist der Instanz-Handle, der von der Funktion

[C16\\_InitInstance\(\)](#) bereitgestellt und an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank übergeben wurde. Zur Identifikation des Feldes wird im Parameter (aFldName) der Name des Feldes angegeben. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_FldInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_FldInfo)).

## Kontakt

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_FldInfoExt(const vHANDLE**

**aInstHdl,**  
**const vINT aFileNo,**  
**const vINT aSbrNo,**  
**const vINT aFldNo,**  
**vC16\_FldInfoExt\* aInfoBlock) :**

**vERROR**

**Informationen über ein Feld ermitteln**

**aInstHdl** Instanz-Handle

**aFileNo** Dateinummer

**aSbrNo** Teildatensatznummer

**aFldNo** Feldnummer

**aInfoBlock** Informationsstruktur

**Resultat** vERROR Fehlerwert

**Befehle der**

**Programmierschnittstelle,**

**Siehe** [C16\\_FldInfoExtByName\(\)](#),

[C16\\_FldInfo\(\)](#),

[C16\\_FldInfoByName\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_FldInfoExt mit Informationen zu einem in der Datenstruktur definierten Feld. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT FldFileName; vINT FldSbrNumber; vINT FldNumber; vi
```

**Komponente** Beschreibung

**InfoSize** Größe der Struktur

**FldFileName** Nummer der Datei, in der das Feld definiert ist

**FldSbrNumber** Nummer des Teildatensatzes, in dem das Feld definiert ist

**FldNumber** Feldnummer

**FldLen** Feldlänge (maximale Länge bei alphanumerischen Feldern)

**FldAttribute** Feldattribute - Reserviert für zukünftige Versionen

**FldTyp** Typ entsprechender

C-Typ

1	_TypeAlpha alphanumerisch	vBYTE[FldLen +
---	---------------------------	----------------

2]

2	_TypeDate Datum	vDATE
---	-----------------	-------

4	_TypeWord ganzzahlig kurz	vWORD
---	---------------------------	-------

7	_TypeInt ganzzahlig lang	vLONGs
---	--------------------------	--------

9	_TypeFloat numerisch	vFLOAT
---	----------------------	--------

10	_TypeLogic logisch	vBOOL8
----	--------------------	--------

11	_TypeTime Zeit	vTIME
----	----------------	-------

**FldInputRight** Eingabeberechtigung

**FldOutputRight** Ausgabeberechtigung

**FldStatus** Feldstatus - Reserviert für zukünftige Versionen

**FldName[24]** Feldname

## Kontakt

Der Parameter (aInstHdl) ist der Instanz-Handle, der von der Funktion

C16\_InitInstance() bereitgestellt und an die Funktion C16\_OpenArea() zum Öffnen der Datenbank übergeben wurde. Zur Identifikation des Feldes werden in den Parametern (aFileNo), (aSbrNo) und (aFldNo) die Datei-, die Teildatensatz- und die Feldnummer angegeben. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_FldInfoExt übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_FldInfoExt)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_FldInfoExtByName(const vHANDLE aInstHdl, const vCHAR\* aFldName, vC16\_FldInfoExt\* aInfoBlock) :**  
vERROR  
Informationen über ein Feld ermitteln  
aInstHdl Instanz-Handle  
aFldName Feldname  
aInfoBlock Informationsstruktur

Resultat vERROR

Fehlerwert

Befehle der

Programmierschnittstelle,  
**Siehe** [C16\\_FldInfoExt\(\)](#),  
[C16\\_FldInfo\(\)](#),  
[C16\\_FldInfoByName\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_FldInfoExt mit Informationen zu einem in der Datenstruktur definierten Feld. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT FldFileName; vINT FldSbrNumber; vINT FldNumber; vi
```

**Komponente** Beschreibung

InfoSize Größe der Struktur

FldFileName Nummer der Datei, in der das Feld definiert ist FldSbrNumber Nummer des Teildatensatzes, in dem das Feld definiert ist

FldNumber Feldnummer

FldLen Feldlänge (maximale Länge bei alphanumerischen Feldern)

FldAttribute Feldattribute - Reserviert für zukünftige Versionen

FldTyp Typ entsprechender

C-Typ

1	_TypeAlpha	alphanumerisch	vBYTE[FldLen + 2]
---	------------	----------------	-------------------

2	_TypeDate	Datum	vDATE
---	-----------	-------	-------

4	_TypeWord	ganzzahlig kurz	vWORD
---	-----------	-----------------	-------

7	_TypeInt	ganzzahlig lang	vLONGs
---	----------	-----------------	--------

9	_TypeFloat	numerisch	vFLOAT
---	------------	-----------	--------

10	_TypeLogic	logisch	vBOOL8
----	------------	---------	--------

11	_TypeTime	Zeit	vTIME
----	-----------	------	-------

FldInputRight Eingabeberechtigung

FldOutputRight Ausgabeberechtigung

FldStatus Feldstatus - Reserviert für zukünftige Versionen

FldName[24] Feldname

Der Parameter (aInstHdl) ist der Instanz-Handle, der von der Funktion

[C16\\_InitInstance\(\)](#) bereitgestellt und an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank übergeben wurde. Zur Identifikation des Feldes wird im Parameter (aFldName) der Name des Feldes angegeben. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_FldInfoExt übergeben. Die Strukturkomponente

## **Kontakt**

**InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden  
(`sizeof(vC16_FldInfoExt)`).**

**Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).**

## Kontakt

**C16\_KeyInfo**(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aKeyNo,  
vC16\_KeyInfo\* aInfoBlock) : vERROR  
Informationen über einen Schlüssel ermitteln  
aInstHdl      Instanz-Handle  
aFileNo      Dateinummer  
aKeyNo      Schlüsselnummer  
aInfoBlock Informationsstruktur  
Resultat      vERROR      Fehlerwert  
                Befehle der  
                Programmierschnittstelle,  
Siehe            C16\_KeyInfoByName(),  
                C16\_KeyFldInfo()

Der Befehl füllt eine Struktur vom Typ vC16\_KeyInfo mit Informationen zu einem in der Datenstruktur definierten Schlüssel. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT KeyFileName; vINT KeyNumber; vINT KeyFldCount; vBO
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
KeyFileName	Nummer der Datei, in welcher der Schlüssel definiert ist
KeyNumber	Nummer des Schlüssels
KeyFldCount	Anzahl der Schlüsselfelder
KeyIsUnique	0 = Schlüssel ist nicht eindeutig, 1 = Schlüssel ist eindeutig
KeyName[24]	Name des Schlüssels

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. In den Parametern (aFileNo) und (aKeyNo) müssen sich die Datei- und die Schlüsselnummer für den Schlüssel befinden, von dem die Informationen abgefragt werden sollen. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_KeyInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_KeyInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_KeyInfoByName(const vHANDLE  
aInstHdl,  
const vCHAR\* aKeyName,  
vC16\_KeyInfo\* aInfoBlock) : vERROR**  
Informationen über einen Schlüssel ermitteln  
aInstHdl Instanz-Handle  
aKeyName Schlüsselname  
aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der  
Programmierschnittstelle,  
Siehe [C16\\_KeyInfo\(\)](#),  
[C16\\_KeyFldInfo\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_KeyInfo mit Informationen zu einem in der Datenstruktur definierten Schlüssel. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT KeyFileName; vINT KeyNumber; vINT KeyFldCount; vBO
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
KeyFileName	Nummer der Datei, in welcher der Schlüssel definiert ist
KeyNumber	Nummer des Schlüssels
KeyFldCount	Anzahl der Schlüsselfelder
KeyIsUnique	0 = Schlüssel ist nicht eindeutig, 1 = Schlüssel ist eindeutig
KeyName[24]	Name des Schlüssels

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. In dem Parameter (aKeyName) muss sich der Schlüsselname für den Schlüssel befinden, von dem die Informationen abgefragt werden sollen. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_KeyInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_KeyInfo)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_KeyFldInfo(const vHANDLE aInstHdl,**  
**const vINT aFileNo,**  
**const vINT aKeyNo,**  
**const vINT aKeyFldNo,**  
**vC16\_KeyFldInfo\* aInfoBlock) : vERROR**  
Informationen über ein Schlüsselfeld ermitteln  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aKeyNo Schlüsselnummer  
aKeyFldNo Nummer des Schlüsselfeldes

aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der Programmierschnittstelle,  
Siehe [C16\\_KeyInfo\(\)](#),  
[C16\\_KeyInfoByName\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_KeyFldInfo mit Informationen zu einem in der Datenstruktur definierten Schlüsselfeld. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT KeyFldFileNumber; vINT KeyFldSbrNumber; vINT KeyFldNu
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
KeyFldFileNumber	Nummer der Datei, in der das Schlüsselfeld definiert ist
KeyFldSbrNumber	Nummer des Teildatensatzes, in dem das Schlüsselfeld definiert

FldTyp	Feldtyp	entsprechender C-Typ
1	_TypeAlpha alphanumerisch	vBYTE[FldLen + 2]
2	_TypeDate Datum	vDATE
4	_TypeWord ganzzahlig kurz	vWORD
7	_TypeInt ganzzahlig lang	vLONGs
9	_TypeFloat numerisch	vFLOAT
10	_TypeLogic logisch	vBOOL8
11	_TypeTime Zeit	vTIME

KeyFldAttributes	Attribute des Schlüsselfeldes	Großschreibung
0x02	_KeyFldAttrUpperCase	Umlaute in alphabetischer Sortierung
0x04	_KeyFldAttrUmlaut	
0x08	_KeyFldAttrSpecialChars ohne Sonderzeichen	
0x10	_KeyFldAttrSoundex1	Soundex Stufe 1
0x20	_KeyFldAttrSoundex2	Soundex Stufe 2
0x40	_KeyFldAttrReverse	absteigende Sortierung

## Kontakt

Die einzelnen Attribute sind gesetzte Bits des Resultats. Um ein einzelnes Attribut abzufragen, wird das Resultat mit einem binären UND kombiniert.

KeyFldMaxLen	Die definierte maximallänge des Schlüsselfeldes
reserved	Nicht benutzt (Alignment-Byte)
KeyFldName[24]	Name des Schlüsselfelds

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. In den Parametern (aFileNo), (aKeyNo) und (aKeyFldNo) müssen sich die Dateinummer, die Schlüsselnummer und die Nummer des Schlüsselfeldes befinden, von dem die Informationen abgefragt werden sollen. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_KeyFldInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_KeyFldInfo)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

```
C16_RecInfo(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aInfoType,  
const vINT aKeyNo,  
vINT* aResult) : vERROR  
Informationen zu einem Datensatz ermitteln  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aInfoType Informationstyp  
aKeyNo Schlüsselnummer  
aResult Zeiger auf das Ergebnis  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle
```



Die Anweisung ist nur noch aus Kompatibilitätsgründen im Sprachumfang enthalten. Es sollte die Methode [C16\\_RecInfo64](#) verwendet werden.

Der Befehl ermittelt Informationen zu einem Datensatz. Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. Im Parameter (aFileNo) wird die Dateinummer angegeben für welche die Informationen abgefragt werden sollen. Als Informationstyp stehen folgende symbolischen Konstanten zur Verfügung:

_RecCount	Mit dieser Option wird die Anzahl der Datensätze in der Datei ermittelt.
_RecID	Das Resultat ist die interne ID des <u>aktuell</u> geladenen Satzes. Das Resultat ist ungültig, wenn kein Satz aus der Datei geladen ist.
_RecSetID	Die <u>Datensatz-ID</u> wird auf den in (aKeyNo) übergebenen Wert gesetzt. Mit diesem Befehl kann die Datensatz-ID zurückgesetzt werden. Das Zurückschreiben eines Datensatzes mit dem Befehl <a href="#">C16_RecReplace</a> erfolgt über die Datensatz-ID. Durch eine Änderung der Datensatz-ID kann der ursprünglich gelesene Datensatz nicht mehr zurückgeschrieben werden.
_RecLen	Die effektive Länge des im Hauptspeicher befindlichen Datensatzes in der Datei wird ermittelt. Das Resultat gibt die Größe des Datensatzes in Bytes zurück. In der effektiven Länge sind alphanumerische Felder nur mit ihrer aktuellen Länge berücksichtigt.
_RecLenPacked	Die gepackte Länge des zuletzt gelesenen (z. B. <a href="#">C16_RecRead</a> ) oder geschriebenen (z. B. <a href="#">C16_RecInsert</a> ) Datensatzes in der Datei wird ermittelt. Änderungen am Feldpuffer werden nicht berücksichtigt. Das Resultat gibt die gepackte Größe des Datensatzes in Bytes zurück.
_RecLockedBy	Wird auf einen gesperrten Datensatz zugegriffen, so kann hiermit die ID-Nummer des Benutzers ermittelt werden, der den Datensatz gesperrt hat.
_RecGetPos	

## Kontakt

**Die Position des aktuellen Satzes innerhalb der Datei (nach Schlüsselsortierung). Die Schlüsselposition eines Satzes gibt allerdings nur die ungefähre Position an. Die Nummer des Schlüssels wird mit (aKeyNo) angegeben.**

**\_RecGetPosReverse** Analog zu **\_RecGetPos**. Die Position des aktuellen Satzes innerhalb der Datei wird jedoch nach umgekehrter Schlüsselsortierung ermittelt.

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

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe **Fehlerwerte**).

## Kontakt

**C16\_RecInfo64**(const vPHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aInfoType,  
const vXLONG aKeyNo,  
vXLONG\* aResult) : vERROR Informationen zu  
einem Datensatz ermitteln aInstHdl Instanz-Handle

aFileNo Dateinummer  
aInfoType Informationstyp  
aKeyNo Schlüsselnummer  
aResult Zeiger auf das Ergebnis  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16 RecInfo

Der Befehl ermittelt Informationen zu einem Datensatz. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Im Parameter (aFileNo) wird die Dateinummer angegeben für welche die Informationen abgefragt werden sollen. Als Informationstyp stehen folgende symbolischen Konstanten zur Verfügung:

<u>_RecCount</u>	Mit dieser Option wird die Anzahl der Datensätze in der Datei ermittelt.
<u>_RecID</u>	Das Resultat ist die interne ID des <u>aktuell geladenen Satzes</u> . Das Resultat ist ungültig, wenn kein Satz aus der Datei geladen ist.
<u>_RecSetID</u>	Die <u>Datensatz-ID</u> wird auf den in (aKeyNo) übergebenen Wert gesetzt. Mit diesem Befehl kann die Datensatz-ID zurückgesetzt werden. Das Zurückschreiben eines Datensatzes mit dem Befehl <u>C16_RecReplace</u> erfolgt über die Datensatz-ID. Durch eine Änderung der Datensatz-ID kann der ursprünglich gelesene Datensatz nicht mehr zurückgeschrieben werden.
<u>_RecLen</u>	Die effektive Länge des im Hauptspeicher befindlichen Datensatzes in der Datei wird ermittelt. Das Resultat gibt die Größe des Datensatzes in Bytes zurück. In der effektiven Länge sind alphanumerische Felder nur mit ihrer aktuellen Länge berücksichtigt.
<u>_RecLenPacked</u>	Die gepackte Länge des zuletzt gelesenen (z. B. <u>C16_RecRead</u> ) oder geschriebenen (z. B. <u>C16_RecInsert</u> ) Datensatzes in der Datei wird ermittelt. Änderungen am Feldpuffer werden nicht berücksichtigt. Das Resultat gibt die gepackte Größe des Datensatzes in Bytes zurück.
<u>_RecLockedBy</u>	Wird auf einen gespererten Datensatz zugegriffen, so kann hiermit die ID-Nummer des Benutzers ermittelt werden, der den Datensatz gesperrt hat.
<u>_RecGetPos</u>	Die Position des aktuellen Satzes innerhalb der Datei (nach Schlüsselsortierung). Die Schlüsselposition eines Satzes gibt allerdings nur die ungefähre Position an. Die Nummer des Schlüssels wird mit (aKeyNo) angegeben.

## Kontakt

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

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

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_LinkInfo**(const vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aLinkNo,  
vC16\_LinkInfo\* aInfoBlock) : vERROR Informationen  
über eine Verknüpfung ermitteln aInstHdl Instanz-  
Handle  
aFileNo Dateinummer  
aLinkNo Verknüpfungsnummer  
aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der Programmierschnittstelle,  
Siehe [C16\\_LinkFldInfo\(\)](#),  
[C16\\_LinkInfoByName\(\)](#)

Der Befehl füllt eine Struktur vom Typ vC16\_LinkInfo mit Informationen zu einer in der Datenstruktur definierten Verknüpfung. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT LinkFileName; vINT LinkNumber; vINT LinkFldCount;
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
LinkFileName	Nummer der Datei, in der die Verknüpfung definiert ist
LinkNumber	Verknüpfungsnummer
LinkFldCount	Anzahl der Verknüpfungsfelder
LinkDestFileName	Nummer der verknüpften Datei
LinkDestKeyNumber	Schlüsselnummer der verknüpften Datei
LinkName[24]	Name der Verknüpfung

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. In den Parametern (aFileNo) und (aLinkNo) müssen sich die Dateinummer und die Nummer der Verknüpfung befinden, von dem die Informationen abgefragt werden sollen. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_LinkInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_LinkInfo)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_LinkInfoByName(const vPHANDLE aInstHdl,  
const vCHAR aLinkName,  
vC16\_LinkInfo\* aInfoBlock) : vERROR** Informationen  
über eine Verknüpfung ermitteln aInstHdl Instanz-  
Handle  
aLinkName Verknüpfungsname  
aInfoBlock Informationsstruktur  
Resultat      vERROR      Fehlerwert  
                Befehle der  
Programmierschnittstelle,  
Siehe            C16\_LinkInfo(),  
                C16\_LinkFldInfo()

Der Befehl füllt eine Struktur vom Typ vC16\_LinkInfo mit Informationen zu einer in der Datenstruktur definierten Verknüpfung. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT LinkFileName; vINT LinkNumber; vINT LinkFldCount;
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
LinkFileName	Nummer der Datei, in der die Verknüpfung definiert ist
LinkNumber	Verknüpfungsnummer
LinkFldCount	Anzahl der Verknüpfungsfelder
LinkDestFileName	Nummer der verknüpften Datei
LinkDestKeyNumber	Schlüsselnummer der verknüpften Datei
LinkName[24]	Name der Verknüpfung

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. In dem Parameter (aLinkName) wird der Name der Verknüpfung übergeben, von der die Informationen abgefragt werden soll. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_LinkInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_LinkInfo)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_LinkFldInfo**(const vPHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aLinkNo,  
const vINT aLinkFldNo,  
vC16\_LinkFldInfo\* aInfoBlock) : vERROR Information  
über ein Verknüpfungsfeld ermitteln  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aLinkNo Nummer der Verknüpfung  
aLinkFldNo Nummer des  
Verknüpfungsfeldes  
Zeiger auf  
aInfoBlock Informationsstruktur  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_LinkInfo()

Der Befehl füllt eine Struktur vom Typ **vC16\_LinkFldInfo** mit Informationen zu einem in der Datenstruktur definierten Verknüpfungsfeld. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT LinkFldFileName; vINT LinkFldSbrNumber; vINT LinkFl
```

Komponente	Beschreibung
InfoSize	Größe der Struktur
LinkFldFileName	Nummer der Datei, in der das Verknüpfungsfeld definiert ist
LinkFldSbrNumber	Nummer des Teildatensatzes, in dem das Verknüpfungsfeld definiert ist
LinkFldNumber	Nummer des Verknüpfungsfeldes
LinkFldTyp	Wert Konstante C-Typ
	1 _TypeAlpha vBYTE[FldLen+2]
	2 _TypeDate vDATE
	4 _TypeWord vWORD
	7 _TypeInt vLONGs
	9 _TypeFloat vFLOAT
	10 _TypeLogic vBOOL8
	11 _TypeTime vTIME
LinkFldAttribute	Attribute des Verknüpfungsfeldes
	0x02 _LinkFldAttrUpperCase Großschreibung
	0x04 _LinkFldAttrUmlaut Umlaute in alphabetischer Sortierung
	0x08 _LinkFldAttrSpecialChars ohne Sonderzeichen
	0x10 _LinkFldAttrSoundex1 Soundex Stufe 1
	0x20 _LinkFldAttrSoundex2 Soundex Stufe 2
	0x40 _LinkFldAttrReverse absteigende Sortierung

## Kontakt

	0x80 _LinkFldAttrPostion	nur Zugriffspositionierung
LinkFldMaxLen	Die definierte Maximallänge des Verknüpfungsfeldes	
reserved	Alignmend-Byte	
LinkFldName[24]	Name des Verknüpfungsfeldes	

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. In den Parametern (aFileNo), (aLinkNo) und (aLinkFldNo) müssen sich die Dateinummer, die Nummer der Verknüpfung und die Nummer des Verknüpfungsfeldes befinden, von dem die Informationen abgefragt werden sollen. In (aInfoBlock) wird ein Zeiger auf eine Informationsstruktur vom Typ vC16\_LinkFldInfo übergeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_LinkFldInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

```
C16_RecLinkInfo(const vHANDLE aInstHdl, const  
vINT aSrcFileNo, const vINT aDstFileNo,  
  
const vINT aLinkNo,  
const vINT aInfoType,  
const vINT unused,  
vINT* aResult) : vERROR  
Informationen zu verknüpften Datensätzen ermitteln aInstHdl  
Instanz-Handle  
aSrcFileNo Nummer der Quell-Datei  
aDstFileNo Nummer der Ziel-Datei  
aLinkNo Nummer der Verknüpfung  
aInfoType Informationstyp  
    unbenutzt, muss 0  
unused  
    enthalten  
aResult Zeiger auf das Ergebnis  
Resultat vERROR Fehlerwert  
    Befehle der  
Siehe Programmierschnittstelle
```

Der Befehl ermittelt Informationen zu einem verknüpften Datensatz. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Im Parameter (aScrFileNo) wird die Nummer der Ausgangsdatei, in (aDstFileNo) die Nummer der Zielfile und in (aLinkNo) die Nummer der Verknüpfung angegeben, für welche die Informationen abgefragt werden sollen. Als Informationstyp stehen folgende symbolischen Konstanten zur Verfügung:

<u>_RecCount</u>	Anzahl der verknüpften Datensätze in der Zielfile. Eine eventuelle Zugriffspositionierung wird dabei ignoriert.
<u>_RecGetPos</u>	Die Position des aktuellen Satzes der Zielfile. Der erste verknüpfte Satz hat dabei die Nummer 1. Das Resultat ist -1, wenn der Satz nicht zur verknüpften Menge gehört.
<u>_RecGetPosReverse</u>	Analog zu <u>_RecGetPos</u> . Die Position des aktuellen Satzes innerhalb der Zielfile wird jedoch nach umgekehrter Schlüsselsortierung ermittelt.
<u>_RecCountPos</u>	Anzahl aller verknüpften Datensätze ab der definierten Zugriffsposition. Der per Zugriffspositionierung verknüpfte Satz wird mitgezählt.
<u>_RecCountNext</u>	Die Anzahl verknüpfter Sätze nach dem aktuellen Satz in der Zielfile. Das Resultat ist -1, wenn der Satz nicht zur verknüpften Menge gehört.

Der Parameter (unused) wird nicht verwendet und muss auf 0 gesetzt sein. Das Ergebnis steht in der Variablen, auf die der übergebene Zeiger (aResult) verweist.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_QueryUserInfo(const vHANDLE  
aInstHdl,  
const vINT aUserID,  
const vBOOL aNextID,  
vC16\_UserInfo\* aInfoBlock) : vERROR**

Informationen über Benutzer ermitteln

aInstHdl Instanz-Handle

aUserID Benutzer-ID

true = der nächste

Benutzer wird gelesen,

aNextID false = der Benutzer mit  
der angegebenen ID wird  
gelesen

aInfoBlock Informationsstruktur

Resultat vERROR Fehlerwert

Befehle der

Programmierschnittstelle,

Siehe C16\_QueryAreaInfo(),

C16\_RecInfo(),

C16\_UserClear()

Der Befehl füllt eine Struktur vom Typ vC16\_UserInfo mit Informationen zum Server.

Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vINT ID; vINT Number; vINT Protocol; vLONG FirstReq; vDATE
```

### Komponente Beschreibung

InfoSize Größe der Struktur

ID Bentzer-ID

Number Benutzernummer

Protocol Kommunikationsprotokoll

FirstReq Anzahl der Sekunden seit dem Login

FirstReqDate Datum des Login

FirstReqTime Uhrzeit des Login

LastReq Anzahl der Sekunden seit der letzten Serveranfrage

LastReqDate Datum der letzten Serveranfrage

LastReqTime Uhrzeit der letzten Serveranfrage

Name Benutzername

SysName Rechnername

SysNameIP Host-Name

Address Netzwerkadresse des Rechners

SysAccount Benutzerkonto

NetAccount Netzwerk-Benutzerkonto

Die Funktion liest Informationen aus der Benutzertabelle des CONZEPT 16-Servers aus. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Im Parameter (aUserID) wird die Benutzer-ID des zu lesenden Benutzereintrags angegeben. Mit dem Parameter (aNextID) wird bestimmt,

## Kontakt

ob der nächste Eintrag (true) oder der Eintrag der angegebenen Benutzer-ID (false) gelesen werden soll. Wird in (aUserID) 0 und in (aNextID) true übergeben, werden die Informationen des ersten Benutzers in die Informationsstruktur übertragen. Die Informationen werden in die Informationsstruktur übertragen, deren Adresse in (aInfoBlock) übergeben wurde. Vor dem Aufruf der Funktion muss die Strukturkomponente InfoSize mit der Größe der Struktur initialisiert werden (sizeof(vC16\_UserInfo)).

Die eigene Benutzer-ID kann mit dem Kommando C16\_QueryAreaInfo(), die ID eines sperrenden Benutzers mit C16\_RecInfo() ermittelt werden.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

Beispiel:

```
// Auslesen aller BenutzervC16_UserInfo UserInfo;// Struktur initialisierenUserInfo.InfoSize = si
```

## Kontakt

Externe Windows Programmierschnittstelle - Text-Funktionen Übersicht  
über die Text-Funktionen der Programmierschnittstelle

- [C16\\_TextClose](#)
- [C16\\_TextCopy](#)
- [C16\\_TextCreate](#)
- [C16\\_TextDelete](#)
- [C16\\_TextOpen](#)
- [C16\\_TextRead](#)
- [C16\\_TextReadData](#)
- [C16\\_TextRename](#)
- [C16\\_TextWrite](#)
- [C16\\_TextWriteData](#)

## Kontakt

**C16\_TextOpen(const  
vHANDLE aInstHdl,  
vHANDLE\* aTextHdl) :**  
**vERROR**  
**Textpuffer anlegen**  
**aInstHdl Instanz-Handle**  
**aTextHdl Text-Handle**  
**Resultat vERROR Fehlerwert**  
**Befehle der**  
**Programmierschnittstelle,**

Siehe

**C16\_TextClose(),**  
**C16\_TextReadData()**

Der Befehl erzeugt einen Textpuffer und stellt den dazugehörigen Text-Handle in (aTextHdl). Ein Textpuffer wird für alle weiteren Textoperationen benötigt. Es können beliebig viele Textpuffer angelegt werden. Wird der Textpuffer nicht mehr verwendet, so muss er mit **C16\_TextClose()** wieder entfernt werden.

Im Parameter (aInstHdl) wird der von **C16\_InitInstance()** bereitgestellte Instanz-Handle, der in der Funktion **C16\_OpenArea()** zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe **Fehlerwerte**). Ist der Rückgabewert gleich **C16ERR\_INSTANCE\_HDL\_INVALID**, ist der übergebene (aInstHdl) ungültig.

## Kontakt

```
C16_TextRead(const vPHANDLE  
aTextHdl,  
const vCHAR* aTextName,  
const vFLAGS aFlags,  
vC16_TextInfo* aTextInfo) : vERROR  
Informationen über einen Text ermitteln  
aTextHdl    Text-Handle  
aTextName Name des Textes  
           Optionen der  
aFlags     Zugriffssteuerung  
aTextInfo   Informationsstruktur  
Resultat    vERROR Fehlerwert  
           Befehle der  
Siehe      Programmierschnittstelle,  
           C16_TextWrite()
```

Der Befehl liest Textinformationen in eine Struktur vom Typ **vC16\_TextInfo** ein. Die Struktur ist folgendermaßen definiert:

```
typedef struct{  vINT    InfoSize;  vCHAR  TextName[24];  vCHAR  TextPrivUser[24];  vCHAR  TextCre
```

Komponente	Beschreibung
------------	--------------

InfoSize	Größe der Struktur
----------	--------------------

*Folgende Elemente können nicht verändert werden:*

TextName	Name des Textes
----------	-----------------

TextPrivUser	Privater Benutzername
--------------	-----------------------

TextCreateUser	Benutzer bei der Erstellung	TextModifiedUser
----------------	-----------------------------	------------------

Benutzer bei der letzten Änderung	TextCreateDate	Datum der
-----------------------------------	----------------	-----------

Erstellung	TextCreateTime	Uhrzeit der Erstellung
------------	----------------	------------------------

TextModifiedDate	Datum der letzten Änderung
------------------	----------------------------

TextModifiedTime	Uhrzeit der letzten Änderung
------------------	------------------------------

TextSize	Größe des Textes in Bytes
----------	---------------------------

TextLines	Anzahl der Zeilen
-----------	-------------------

*Folgende Elemente können verändert werden:*

TextGroup	Name der Textgruppe
-----------	---------------------

TextReadPL	Mindest-Leseberechtigung
------------	--------------------------

TextWritePL	Mindest-Schreibberechtigung
-------------	-----------------------------

TextPrivate	Privater Text
-------------	---------------

TextEncrypted	Verschlüsselte Speicherung
---------------	----------------------------

Mit dem Parameter (aTextHdl) oder (aTextName) wird der Text, zu dem die Informationen ermittelt werden sollen, identifiziert. In (aTextHdl) wird entweder ein mit C16\_TextOpen() erzeugter Text-Handle oder NULL übergeben. Wurde NULL übergeben, muss in (aTextName) der Name des Textes angegeben werden. Über den Parameter (aFlags) wird gesteuert, von welchem Text die Informationen ermittelt werden. Folgende symbolische Konstanten stehen zur Verfügung:

## Kontakt

<b>_TextFirst</b>	<b>Der erste Text wird gelesen.</b>
<b>_TextLast</b>	<b>Der letzte Text wird gelesen.</b>
<b>_TextPrev</b>	<b>Der Text mit dem nächstkleineren Namen wird gelesen. Sofern kein weiterer Text vorhanden ist, wird als Resultat _rNoRec zurückgeliefert.</b>
<b>_TextNext</b>	<b>Der Text mit dem nächstgrößeren Namen wird gelesen. Sofern kein weiterer Text vorhanden ist, wird als Resultat _rNoRec zurückgeliefert.</b>
<b>_TextLock</b>	<b>Der Text wird gesperrt. Ist der Text durch einen anderen Benutzer gesperrt, wird als Resultat _rLocked zurückgeliefert.</b>
<b>_TextUnlock</b>	<b>Der Text wird entsperrt.</b>

**Vor dem Funktionsaufruf muss die Strukturkomponente InfoSize mit der Größe der Informationsstruktur initialisiert werden (sizeof(vC16\_TextInfo)). Nach erfolgreicher Ausführung der Funktion stehen die Informationen in der Informationsstruktur (aTextInfo) zur Verfügung.**

**Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:**

<b>_rOk (0)</b>	<b>Der Text wurde gelesen.</b>
<b>_rLocked (1)</b>	<b>Der Text ist gesperrt.</b>
<b>_rNoKey (3)</b>	<b>Der Text ist nicht vorhanden. Es wurde der</b>

**\_rLastRec (4)**

**C16ERR\_TEXT\_HDL\_INVALID  
C16ERR\_TEXT\_NAME\_INVALID**

**C16ERR\_TEXT\_RIGHTS**

**nächst folgende Text gelesen.**

**Der angegebene Text konnte nicht gelesen werden.**

**Es gibt auch keinen nachfolgenden Text.**

**(aTextHdl) ist ungültig.**

**(aTextName) ist zu lang.**

**In (aTextInfo) ist der Wert von InfoSize nicht**

**C16ERR\_INFO\_BLOCKSIZE\_INVALID korrekt.**

**Leseberechtigung nicht ausreichend. In diesem Fall steht der Name des Textes im Feld TextName. Alle anderen Felder von (aTextInfo) sind leer.**

## Kontakt

**C16\_TextWrite(const vHANDLE aTextHdl,  
const vFLAGS aFlags,  
vC16\_TextInfo\* aTextInfo) :**  
**vERROR**  
**Textinformation speichern**  
**aTextHdl Text-Handle**  
    **Text sperren**  
    (\_TextLock),  
aFlags  
    **entsperren**  
    (\_TextUnlock)  
**aTextInfo Informationsstruktur**  
**Resultat vERROR Fehlerwert**  
    **Befehle der**  
**Siehe     Programmierschnittstelle,**  
        C16\_TextRead()

Mit dieser Funktion können die Textinformationen verändert werden. Die Textinformation muss vorher mit C16\_TextRead() gelesen und gesperrt worden sein. Die veränderbaren Felder von (aTextInfo) sind bei C16\_TextRead() aufgeführt. In (aTextHdl) wird der mit C16\_TextOpen() erzeugte Handle eines Textpuffers übergeben. Der Parameter (aFlags) entscheidet, ob der Text nach Ausführung der Funktion weiter gesperrt bleibt (\_TextLock) oder entsperrt wird (\_TextUnlock).

Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

<b>_rOk (0)</b>	<b>Der Text wurde geschrieben.</b>
<b>_rNoLock (7)</b>	<b>Der Text ist nicht gesperrt.</b>
<b>_rDeadlock (10)</b>	<b>Der Text konnte aufgrund einer Verklemmung nicht gespeichert werden.</b>
<b>C16ERR_TEXT_HDL_INVALID</b>	<b>(aTextHdl) ist ungültig.</b>
<b>C16ERR_INFO_BLOCKSIZE_INVALID</b>	<b>In (aTextInfo) ist der Wert von InfoSize nicht korrekt.</b>
<b>C16ERR_TEXT_RIGHTS</b>	<b>Schreibberechtigung nicht ausreichend.</b>
<b>C16ERR_TEXT_GROUP_INVALID</b>	<b>Textgruppe (aTextInfo-&gt;TextGroup) ist länger als 20 Zeichen.</b>

```
C16_TextReadData(const
vPHANDLE aTextHdl,
const void* aBuffer,
const vFLAGS aFlags,
const vINT aBufferSize,
vINT* aTextSize,
vINT* aTextLines) : vERROR
```

Text lesen

aTextHdl	Text-Handle zu füllender
aBuffer	Textpuffer

aFlags	Optionen Größe des
--------	-----------------------

**aBufferSize Zielpuffers**

Anzahl der

aTextSize übertragenen

Zeichen

aTextLines	Anzahl der
------------	------------

übertragenen Zeilen

Resultat vERROR Fehlerwert

Befehle der  
Programmierschnittstelle,

Siehe

C16\_TextOpen(),

C16\_TextWriteData()

Mit dieser Funktion wird der Inhalt eines Textes gelesen. Vorher muss die Textinformation mit C16\_TextRead() gelesen worden sein. Der Textinhalt kann entweder komplett oder in einzelnen Teilen übertragen werden. Zur kompletten Übertragung muss (aBuffer) ausreichend groß sein. Ist (aBuffer) kleiner als die Gesamtgröße des Textes, so kann C16\_TextReadData() mehrfach aufgerufen werden, wobei ab dem zweiten Aufruf die Konstante \_TextAppend verwendet werden muss. Durch die Verwendung von \_TextBinary können beliebige, in Texten gespeicherte Binärdaten gelesen werden.

In (aTextHdl) wird der mit C16\_TextOpen() erzeugte Handle eines Textpuffers übergeben, von dem der Textinhalt gelesen werden soll. In den Parametern (aBuffer) und (aBufferSize) wird ein Zeiger auf einen Puffer, in den der Textinhalt übertragen wird und dessen Größe übergeben. Um die Übertragung zu steuern stehen für den Parameter (aFlags) folgende symbolischen Konstanten zur Verfügung:

_TextAppend	Es werden die Daten ab dem Ende der letzten Übertragung gelesen.
_TextSingleLine	Es wird genau eine Zeile übertragen.
_TextBinary	Die Daten werden binär (1:1) übertragen.
_TextEolNull	Die Zeilen werden mit NUL (ASCII 0) beendet.
_TextEolCR	Die Zeilen werden mit CR (ASCII 13) beendet.
_TextEolLF	Die Zeilen werden mit LF (ASCII 10) beendet.
_TextEolCRLF	Die Zeilen werden mit CR+LF (ASCII 13+10) beendet.
_Text_CharSet_C16	Es findet keine Zeichensatzwandlung statt.

## Kontakt

**\_Text\_CharSet\_OEM** Der Text wird in den OEM-Zeichensatz umgewandelt.

**\_Text\_CharSet\_ANSI** Der Text wird in den ANSI-Zeichensatz umgewandelt.

In den Parametern (aTextSize) und (aTextLines) werden die Anzahl der gelesenen Zeichen bzw. die Anzahl der gelesenen Zeilen abgelegt. Werden diese Daten nicht benötigt, kann NULL übergeben werden.

Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

**\_rOk (0)** Der Textinhalt wurde übertragen.

**C16ERR\_TEXT\_HDL\_INVALID (aTextHdl)** ist ungültig.

**C16ERR\_TEXT\_UNDEFINED** Im Textpuffer steht keine gültige Textinformation.

## Kontakt

**C16\_TextWriteData**(const  
vHANDLE aTextHdl,  
const void\* aBuffer,  
const vFLAGS aFlags,  
const vINT aTextSize) : vERROR  
Text speichern  
aTextHdl Text-Handle  
aBuffer Zeiger auf Quellpuffer  
aFlags Optionen  
aTextSize Anzahl der zu  
schreibenden Bytes  
Resultat vERROR Fehlerwert  
**Siehe** [Befehle der Programmierschnittstelle](#),  
[C16\\_TextOpen\(\)](#),  
[C16\\_TextReadData\(\)](#)

Mit dieser Funktion kann der Inhalt eines Textes geschrieben werden. Der Text muss vorher mit **C16\_TextRead()** gelesen und gesperrt worden sein. Alternativ kann ein neuer Text mit **C16\_TextCreate()** angelegt und gesperrt werden. Der Text kann entweder komplett oder in mehreren Teilen übertragen werden. Bei der Übertragung mehrerer Teile muss bei allen Aufrufen von **C16\_TextWriteData()** die Konstante **\_TextAppend** verwendet werden, ausgenommen beim letzten Aufruf. In (aTextHdl) wird der mit **C16\_TextOpen()** erzeugte Handle eines Textpuffers übergeben, von dem der Textinhalt geschrieben werden soll. In den Parametern (aBuffer) und (aTextSize) wird der Zeiger auf einen Puffer, in dem sich die zu schreibenden Daten befinden und die Anzahl der Bytes in diesem Puffer übergeben. Um die Übertragung zu steuern, stehen für den Parameter (aFlags) folgende symbolischen Konstanten zur Verfügung:

<b>_TextAppend</b>	Es werden die Daten ab dem Ende der letzten Übertragung gelesen.
<b>_TextBinary</b>	Die Daten werden binär (1:1) übertragen.
<b>_TextEoNull</b>	Die Zeilen werden mit NUL (ASCII 0) beendet.
<b>_TextEoCR</b>	Die Zeilen werden mit CR (ASCII 13) beendet.
<b>_TextEoLF</b>	Die Zeilen werden mit LF (ASCII 10) beendet.
<b>_TextEoCRLF</b>	Die Zeilen werden mit CR+LF (ASCII 13+10) beendet.
<b>_Text_CharSet_C16</b>	Es findet keine Zeichensatzwandlung statt.
<b>_Text_CharSet_OEM</b>	Der Text wird in den OEM-Zeichensatz umgewandelt.
<b>_Text_CharSet_ANSI</b>	Der Text wird in den ANSI-Zeichensatz umgewandelt. Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

<b>_rOk (0)</b>	Der Textinhalt wurde geschrieben.
<b>_rNoLock (7)</b>	Der Text ist nicht gesperrt.
<b>C16ERR_TEXT_HDL_INVALID (aTextHdl)</b>	ist ungültig.
<b>C16ERR_TEXT_RIGHTS</b>	Schreibberechtigung nicht ausreichend.

## Kontakt

**C16\_TextClose(const  
vHANDLE aTextHdl) : vERROR**  
Textpuffer entfernen  
aTextHdl Text-Handle  
Resultat vERROR Fehlerwert

Befehle der

Siehe Programmierschnittstelle,  
C16\_TextOpen()

Der Befehl entfernt einen Textpuffer, der mit C16\_TextOpen() erzeugt wurde. Der Handle des Puffers ist anschließend nicht mehr gültig.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_TextCreate(const  
vHANDLE aTextHdl,  
const vCHAR\* aTextName,  
const vBOOL aLockText) :**  
**vERROR**  
**Text erzeugen**  
**aTextHdl** Text-Handle  
**aTextName** Textname  
aLockText Text sperren (true),  
nicht sperren (false)  
**Resultat** vERROR Fehlerwert  
**Befehle der**  
**Siehe** Programmierschnittstelle,  
C16\_TextDelete()

Der Befehl erzeugt einen leeren Text in der Datenbank. Es darf noch kein Text mit dem angegebenen Namen vorhanden sein. Der Name des Textes darf maximal 20 Zeichen lang sein. Wird im Parameter (aLockText) true übergeben, wird der Text gleichzeitig gesperrt.

Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

<b>_rOk (0)</b>	Der Text wurde angelegt.
<b>_rExists (6)</b>	Es existiert bereits ein Text gleichen Namens.
<b>_rDeadlock (10)</b>	Der Text konnte aufgrund einer Verklemmung nicht erzeugt werden.
<b>C16ERR_TEXT_HDL_INVALID</b>	(aTextHdl) ist ungültig.
<b>C16ERR_TEXT_NAME_INVALID</b>	(aTextName) ist ungültig.

## Kontakt

**C16\_TextCopy(const vHANDLE  
aTextHdl,  
const vCHAR\* aTextName,  
const vCHAR\* aTextNameNew)  
: vERROR Text**  
kopieren  
**aTextHdl**Text-Handle  
**aTextName** Textname **aTextNameNew**  
Neuer Textname  
Resultat            vERROR Fehlerwert  
                  Befehle der  
Siehe             Programmierschnittstelle,  
                  C16\_TextRename()

Der Befehl kopiert einen Text in der Datenbank. Es darf noch kein Text mit dem Namen der Kopie in der Datenbank vorhanden sein. Der Benutzer muss über eine ausreichende Leseberechtigung für den Ursprungstext verfügen. In (aTextHdl) wird der mit C16\_TextOpen() erzeugte Handle eines Textpuffers oder in (aTextName) der Name eines Textes (aTextHdl ist dann NULL) übergeben, der kopiert werden soll. (aTextNameNew) ist der Name des neuen Textes (maximal 20 Zeichen). Wird sowohl (aTextHdl), als auch (aTextName) angegeben, hat der Handle vorrang.

Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

<u>_rOk (0)</u>	Der Text wurde kopiert.
<u>_rLocked (1)</u>	Der Ursprungstext ist gesperrt.
<u>_rNoKey (3)</u>	Der Ursprungstext ist nicht vorhanden.
<u>_rExists (6)</u>	Es existiert bereits ein Text gleichen Namens.
<u>_rDeadlock (10)</u>	Der Text konnte aufgrund einer Verklemmung nicht kopiert werden.
<b>C16ERR_TEXT_HDL_INVALID</b>	(aTextHdl) ist ungültig.
<b>C16ERR_TEXT_NAME_INVALID</b>	(aTextName) oder (aTextnameNew) ist ungültig.
<b>C16ERR_TEXT_RIGHTS</b>	Die Berechtigung ist nicht ausreichend.

## Kontakt

**C16\_TextRename(const  
vHANDLE aTextHdl,  
const vCHAR\* aTextName,  
const vCHAR\* aTextNameNew)  
: vERROR**

**Text umbenennen**  
**aTextHdl** Text-Handle  
**aTextName** Textname  
**aTextNameNew** Neuer Textname  
**Resultat** vERROR Fehlerwert  
**Befehle der**  
**Siehe** Programmierschnittstelle,  
C16\_TextCopy()

Ein Text in der Datenbank wird umbenannt. Es darf noch kein Text mit dem neuen Namen vorhanden sein. Der Benutzer muss über eine ausreichende Schreibberechtigung für den Text verfügen. In (aTextHdl) oder (aTextName) wird der Text übergeben, der umbenannt werden soll. Wird der Text über seinen Namen angesprochen, muss in (aTextHdl) NULL übergeben werden.

Im Parameter (aTextNameNew) steht der neue Name des Textes (maximal 20 Stellen).

Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

<b>_rOk (0)</b>	Der Text wurde umbenannt.
<b>_rLocked (1)</b>	Der Text ist gesperrt.
<b>_rNoKey (3)</b>	Der Text ist nicht vorhanden.
<b>_rExists (6)</b>	Es existiert bereits ein Text gleichen Namens.
<b>C16ERR_TEXT_HDL_INVALID</b>	(aTextHdl) ist ungültig.
<b>C16ERR_TEXT_NAME_INVALID</b>	(aTextName) oder (aTextNameNew) ist ungültig.
<b>C16ERR_TEXT_RIGHTS</b>	Die Berechtigung ist nicht ausreichend.

## Kontakt

**C16\_TextDelete(const  
vHANDLE aTextHdl, const  
vCHAR\* aTextName) : vERROR**  
Text löschen  
aTextHdl Text-Handle  
aTextName Textname  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_TextCreate()

Der Befehl löscht einen Text in der Datenbank. Der Benutzer muss über eine ausreichende Schreibberechtigung für den Text verfügen. Der zu löschen Text wird entweder über den Text-Handle (aTextHdl) oder über seinen Namen (aTextName) identifiziert. Werden beide Parameter angegeben, wird der Text im Textpuffer gelöscht.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

Folgende Fehlerwerte vom Typ vERROR können zurückgegeben werden:

<b>_rOk (0)</b>	Der Text wurde gelesen.
<b>_rLocked (1)</b>	Der Text ist gesperrt.
<b>_rNokey (3)</b>	Der Text ist nicht vorhanden.
<b>C16ERR_TEXT_HDL_INVALID</b>	(aTextHdl) ist ungültig.
<b>C16ERR_TEXT_NAME_INVALID</b>	(aTextName) ist ungültig.
<b>C16ERR_TEXT_RIGHTS</b>	Die Berechtigung ist nicht ausreichend.

## Kontakt

Externe Programmierschnittstelle - Funktionen für Selektionen Übersicht über die Selektions-Funktionen der Programmierschnittstelle

- C16\_SelClear
- C16\_SelClose
- C16\_SelOpen
- C16\_SelRead
- C16\_SelRecDelete
- C16\_SelRecInsert

## Kontakt

**C16\_SelOpen(const vHANDLE  
aInstHdl, vPHandle\* aSelHdl) :**  
**vERROR**  
Selektionspuffer anlegen  
aInstHdl Instanz-Handle  
aSelHdl Selektionsdeskriptor  
Resultat vERROR Fehlerwert

Siehe [Befehle der  
Programmierschnittstelle,](#)  
[C16\\_SelClose\(\),](#)  
[C16\\_SelRead\(\)](#)

Diese Funktion legt einen neuen Selektionspuffer an, über den auf den auf Selektionen und deren Menge zugegriffen werden kann.

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben.

Der erzeugte Deskriptor wird in (aSelHdl) zurückgegeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_SelRead(const vPHANDLE aSelHdl,  
const vINT aFileNo,  
const vCHAR\* aSelName,  
const vFLAGS aFlags,  
vC16\_SelInfo\* aInfo) : vERROR**  
Selektion in den Puffer laden  
aSelHdl Selektionsdeskriptor  
aFileNo Dateinummer  
aSelName Name der Selektion  
aFlags Optionen  
Informationen zur  
aInfo Selektion

**Resultat vERROR Fehlerwert**  
Befehle der  
**Siehe Programmierschnittstelle,  
C16\_SelOpen()**

Mit dieser Funktion wird eine Selektion in den Puffer geladen. Der Puffer muss zuvor mit dem Befehl **C16\_SelOpen** eingerichtet werden. Sofern die angegebene Selektion nicht vorhanden ist, wird die Selektion mit dem nächstgrößeren Namen geladen und **\_rNoKey** zurückgeliefert.

Im Parameter (aSelHdl) wird der von **C16\_SelOpen()** bereitgestellte Selektionsdeskriptor übergeben.

In (aFileNo) wird die Nummer der Datei angegeben, in der die Selektion definiert ist.

Der Name der Selektion in (aSelName) kann entfallen (NULL), wenn **\_SelFirst**, **\_SelLast**, **\_SelPrev** oder **\_SelNext** in (aFlag) angegeben wird, oder die zuletzt gelesene Selektion erneut gelesen werden soll.

In (aFlag) können folgende Optionen angegeben werden:

**\_SelFirst** Die erste Selektion der Datei wird geladen.  
**\_SelLast** Die letzte Selektion der Datei wird geladen.  
**\_SelPrev** Die Selektion mit dem nächstkleineren Namen wird geladen. Sofern keine weitere Selektion in der Datei vorhanden ist, wird als Resultat **\_rNoRec** zurückgeliefert.  
**\_SelNext** Die Selektion mit dem nächstgrößeren Namen wird geladen. Sofern keine weitere Selektion in der Datei vorhanden ist, wird als Resultat **\_rNoRec** zurückgeliefert.  
**\_SelLock** Die gelesene Selektion wird gesperrt. Dies ist nur dann der Fall, wenn **\_rOk** zurückgeliefert wird (Selektion vorhanden). Ist die Selektion durch einen anderen Benutzer gesperrt, so ist das Resultat **\_rLocked**.

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

## Kontakt

\_SelUnlock Die gelesene Selektion wird entsperrt.

\_SelKeyMode Alternativer Verarbeitungsmodus

Die Angabe von \_SelLock ist notwendig, wenn mit den Ergebnismengen der Selektion gearbeitet werden soll (C16\_SelClear(), C16\_SelRecInsert(), C16\_SelRecDelete(), C16\_RecRead() und C16\_RecLink()).

Für das Lesen von Datensätzen werden die Funktionen C16\_RecRead() bzw.

C16\_RecLink() verwendet, wobei anstelle der Schlüssel- bzw. Verknüpfungsnummer der Deskriptor des Selektionspuffers angegeben wird. Dabei muss ein Cast des Deskriptors auf vINT verwendet werden.

Falls (aInfo) nicht NULL ist, werden verschiedene Informationen zu der Selektion in einer Struktur zurückgegeben. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_SelInfo)). Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT   InfoSize;  vCHAR  SelName[24];  vCHAR  SelModifiedUser[24];  vDATE  SelCr
```

**Komponente** Beschreibung

**InfoSize** Größe der Struktur in Byte

**SelName** Selektionsname

**SelModifiedUser** Benutzer bei der letzten Änderung

**SelCreateDate** Datum der Erstellung **SelCreateTime** Uhrzeit

der Erstellung **SelModifiedDate** Datum der letzten Änderung

**SelModifiedTime** Uhrzeit der letzten Änderung

**SelExecuteDate** Datum der letzten Durchführung

**SelExecuteTime** Uhrzeit der letzten Durchführung

**SelModifyPL** Berechtigung für Änderung

**SelExecutePL** Berechtigung für Durchführung

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_SelClear(const vHANDLE**

**aSelHdl) : vERROR**

Inhalt des Selektionspuffers löschen

aSelHdl Selektionsdeskriptor

Resultat vERROR Fehlerwert

Befehle der

Siehe Programmierschnittstelle,

C16\_SelRead()

Mit dieser Funktion wird der Inhalt des Selektionspuffers komplett gelöscht. Eine Selektion muss vorher mit C16\_SelRead() sperrend gelesen worden sein. Nach

**C16\_SelClear()** verfügt die Selektion über eine leere Ergebnismengen. Dadurch kann beispielsweise die Ergebnismenge mit C16\_SelRecInsert() gefüllt werden.

Im Parameter (aSelHdl) wird der von C16\_SelOpen() bereitgestellte

Selektionsdeskriptor übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen

Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_SelRecInsert(const vPHANDLE  
aSelHdl, const vINT aFileNo) :**  
**vERROR**  
Datensatz der Selektion hinzufügen  
aSelHdl Selektionsdeskriptor  
aFileNo Dateinummer  
Resultat vERROR Fehlerwert

Siehe **Befehle der  
Programmierschnittstelle,**  
**C16 SelRead(),**  
**C16 SelRecDelete()**

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

Im Parameter (aSelHdl) wird der von **C16 SelOpen()** bereitgestellte Selektionsdeskriptor übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe **Fehlerwerte**).

## Kontakt

**C16\_SelRecDelete(const vHANDLE  
aSelHdl, const vINT aFileNo) :**  
**vERROR**  
Datensatz aus der Selektion entfernen  
aSelHdl Selektionsdeskriptor  
aFileNo Dateinummer  
Resultat vERROR Fehlerwert

Siehe **Befehle der  
Programmierschnittstelle,**  
**C16 SelRead(),**  
**C16 SelRecInsert()**

Diese Funktion entfernt den aktuell im Hauptspeicher befindlichen Datensatz der Datei (aFileNo) aus einer Ergebnismenge der Selektion im Puffer. Dabei ist zu beachten, dass der Sortierungswert des Satzes in der Ergebnismenge mit den tatsächlichen Sortierungswerten im Datensatz übereinstimmen muss, da sonst der Datensatz nicht gelöscht werden kann (beispielsweise ist die Selektionsmenge nach Namen sortiert, der Name im Datensatz hat sich mittlerweile aber verändert).

**C16\_SelRecDelete()** kann sowohl bei der Hauptergebnismenge als auch bei verknüpften Ergebnismengen benutzt werden. Für die Hauptergebnismenge kann auch 0 in (aFileNo) übergeben werden.

Im Parameter (aSelHdl) wird der von **C16\_SelOpen()** bereitgestellte Selektionsdeskriptor übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe **Fehlerwerte**).

## Kontakt

**C16\_SelClose(const vHANDLE  
aSelHdl) : vERROR**  
Selektionspuffer entfernen  
aSelHdl Selektionsdeskriptor  
Resultat vERROR Fehlerwert

Befehle der  
Siehe Programmierschnittstelle,  
C16 SelOpen()

Mit dieser Funktion wird der durch C16 SelOpen() erzeugte Selektionspuffer wieder entfernt.

Im Parameter (aSelHdl) wird der von C16 SelOpen() bereitgestellte Selektionsdeskriptor übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

Externe Programmierschnittstelle - Funktionen für binäre Objekte

Übersicht über die Funktionen für binäre Objekte der Programmierschnittstelle

- [C16\\_BinClose](#)
- [C16\\_BinCopy](#)
- [C16\\_BinDelete](#)
- [C16\\_BinDirRead](#)
- [C16\\_BinInfo](#)
- [C16\\_BinMove](#)
- [C16\\_BinOpen](#)
- [C16\\_BinReadToFile](#)
- [C16\\_BinReadToMem](#)
- [C16\\_BinRename](#)
- [C16\\_BinUpdate](#)
- [C16\\_BinWriteFromFile](#)
- [C16\\_BinWriteFromMem](#)

## Kontakt

**C16\_BinClose(const vHANDLE aHdl)**

: vERROR

**Binäres Objekt / Verzeichnis schließen**

Deskriptor des Objektes /

aHdl

Verzeichnisses

**Resultat vERROR      Fehlerwert**

Befehle der

**Siehe      Programmierschnittstelle,**

**C16\_BinOpen()**

Mit dieser Funktion wird das binäre Objekt bzw. Verzeichnis (aHdl) geschlossen und entsperrt. Der Deskriptor ist anschließend nicht mehr gültig.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_BinCopy(const vHANDLE  
aBinHdlSource, const vHANDLE  
aBinHdlDestination) : vERROR**  
Inhalt eines binären Objektes kopieren  
Deskriptor des  
aBinHdlSource  
Quellobjektes  
Deskriptor des  
aBinHdlDestination  
Zielobjektes  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_BinMove()

Mit dieser Funktion wird der Inhalt des Ausgangsobjekts (aBinHdlSource) in das Zielobjekt (aBinHdlDestination) übertragen. Das Zielobjekt muss dazu exklusiv gesperrt sein (\_BinLock bzw. \_BinSingleLock).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden positiven Fehlerwerte:

\_rNoLock Das Zielobjekt ist nicht exklusiv gesperrt.  
\_rDeadlock Verklemmung aufgetreten.

## Kontakt

**C16\_BinDelete(const vPHANDLE aInstHdl, const vPHANDLE aDirHdl, const vCHAR\* aName, vFLAGS aFlags) :**  
**vERROR**  
Binäres Objekt oder Verzeichnis löschen  
aInstHdl Instanz-Handle  
    Deskriptor des  
aDirHdl Elternverzeichnisses  
aName Objektname  
    Optionen  
        **\_BinClearOnly** Objektinhalt  
            löschen  
aFlags    **\_BinDirectory**   Verzeichnis löschen  
          **\_BinDeleteAll**   Verzeichnis und  
                          Unterverzeichnisse  
                          löschen  
Resultat vERROR      Fehlerwert

Siehe     Befehle der Programmierschnittstelle,  
          C16\_BinOpen()

Mit dieser Funktion wird ein binäres Objekt gelöscht.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

In (aDirHdl) wird der Deskriptor des Ausgangsverzeichnisses angegeben.



Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (aDirHdl) 0 angegeben. Das Wurzelverzeichnis ist immer vorhanden und braucht auch nicht geöffnet zu werden.

Folgende Optionen (aFlags) können angegeben werden:

- **\_BinClearOnly**

Der Inhalt des Objekts (aName) wird gelöscht.

- **\_BinDirectory**

Das Verzeichnis (aName) wird gelöscht.

- **\_BinDeleteAll**

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

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden Fehlerwerte:

<b>_rLocked</b>	Das Objekt (aName) ist gesperrt.
<b>_rDeadlock</b>	Verklemmung aufgetreten.

## **Kontakt**

**C16ERR\_BIN\_NAME\_INVALID** Objektname (aName) ist ungültig.

**C16ERR\_BIN\_NO\_PATH** Das Objekt (aName) existiert nicht.

## Kontakt

**C16\_BinDirRead(const vPHANDLE  
aInstHdl, const vPHANDLE aDirHdl,  
const vCHAR\* aName, vFLAGS aFlags,  
vC16\_BinObj\* aBinData) : vERROR**

Binäres Verzeichnis lesen

aInstHdl Instanz-Handle

aDirHdl	Deskriptor des Ausgangsverzeichnisses	
aName	Referenzeintrag	
	Optionen	
0	Das angegebene Objekt lesen	
_BinFirst	Ersten Eintrag lesen	
_BinLast	Letzten Eintrag lesen	
aFlags	_BinNext	Eintrag nach Referenzeintrag lesen
	_BinPrev	Eintrag vor Referenzeintrag lesen
	_BinDirectory	Unterverzeichnisse lesen

aBinData Daten des gelesenen binären Objektes / Verzeichnisses

Resultat vERROR Fehlerwert

Befehle der  
Siehe [Programmierschnittstelle](#),  
[C16\\_BinOpen\(\)](#)

Mit dieser Funktion wird ein Verzeichnis von binären Objekten geöffnet oder neu angelegt.

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben.

In (aDirHdl) wird der Deskriptor des Ausgangsverzeichnisses angegeben. Das Ausgangsverzeichnis (aDirHdl) muss mit [C16\\_BinOpen\(..., \\_BinDirectory\)](#) geöffnet worden sein.



Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (aDirHdl) 0 angegeben. Das Wurzelverzeichnis ist immer vorhanden und braucht auch nicht geöffnet zu werden.

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

- 0

## Kontakt

Der Verzeichniseintrag mit dem in (aName) angegebenem Namen wird gelesen. Ist kein Verzeichniseintrag mit dem Namen vorhanden, wird der Eintrag mit dem nächst höheren Namen gelesen. Ist kein nächst höherer vorhanden, wird \_rNoRec zurückgegeben.

- **\_BinFirst**

Der erste Verzeichniseintrag wird gelesen.

- **\_BinLast**

Der letzte Verzeichniseintrag wird gelesen.

- **\_BinNext**

Der Verzeichniseintrag nach dem Referenzeintrag (aName) wird gelesen.

- **\_BinPrev**

Der Verzeichniseintrag vor dem Referenzeintrag (aName) wird gelesen.

- **\_BinDirectory**

Die Unterverzeichnisse werden gelesen.

Konnte kein Eintrag gelesen werden (zum Beispiel, weil bei \_BinNext kein Folgeeintrag existiert), wird als Ergebnis \_rNoRec zurückgegeben.

Falls (aBinData) nicht NULL ist, werden verschiedene Informationen zu dem binären Objekt bzw. Verzeichnis in einer Struktur zurückgegeben. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vXLONG ID; vXLONG StorageID; vCHAR Name[64]; vCHAR FullName[512]; vCHA
```

Komponente	Beschreibung
ID	Identität des Objektes / Verzeichnisses
StorageID	Storage-ID (nur bei binären Objekten)
Name	Name des Objektes / Verzeichnisses
FullName	Pfad und Name des Objektes / Verzeichnisses (nur verfügbar nach <u>C16_BinInfo()</u> mit aQueryPath = TRUE)
Custom	Benutzerdefinierte Eigenschaft
Created	Erstellungszeitpunkt des Objektes
Modified	Zeitpunkt der letzten Änderung
TimeExternal	Datum und Uhrzeit der letzten Änderung der externen Datei vor Import
CreatedUser	Benutzer, der das Objekt erzeugt hat
ModifiedUser	Benutzer, der das Objekt zuletzt geändert hat
TypeUser	Benutzerdefinierte Typinformation
TypeMIME	MIME-Typ des Objektes
SizeDba	Speicherverbrauch des Objektes in der Datenbank
SizeOrg	Originalgröße des Objektes
Compression	Kompressionsstufe

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen

## **Kontakt**

**negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden positiven Fehlerwerte:**

**\_rNoRec Das Verzeichnis existiert nicht. Kein weiteres Verzeichnis vorhanden.**

## Kontakt

**C16\_BinInfo(const vPHANDLE aHdl, const vBOOL aQueryPath,  
vC16\_BinObj\* aBinData) : vERROR**

Informationen eines binären Objektes / Verzeichnisses ermitteln

Deskriptor des binären  
aHdl Objektes / Verzeichnisses

**aQueryPath** Eintrag FullName auch  
füllen

Daten des gelesenen  
aBinData binären Objektes /

Verzeichnisses

**Resultat** vERROR Fehlerwert  
Befehle der

**Siehe** Programmierschnittstelle,  
C16\_BinOpen()

Mit dieser Funktion können Informationen eines binären Objektes bzw. Verzeichnisses ermittelt werden.

In (aHdl) wird der Deskriptor des binären Objektes bzw. Verzeichnisses angegeben.

Mit (aQueryPath) kann angegeben werden, ob auch der Eintrag FullName ermittelt wird. Dieser muss unter Umständen aus der Datenbank ermittelt werden. Alle anderen Daten stehen beim Lesen bereits im Speicher.

Falls (aData) nicht NULL ist, werden verschiedene Informationen zu dem binären Objekt bzw.

Verzeichnis in einer Struktur zurückgegeben. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vXLONG ID; vXLONG StorageID; vCHAR Name[64]; vCHAR FullName[512]; vCHA
```

### Komponente Beschreibung

**ID** Identität des Objektes / Verzeichnisses

**StorageID** Storage-ID (nur bei binären Objekten)

**Name** Name des Objektes / Verzeichnisses

**FullName** Pfad und Name des Objektes / Verzeichnisses (nur verfügbar, wenn das Argument aQueryPath = TRUE ist)

**Custom** Benutzerdefinierte Eigenschaft

**Created** Erstellungszeitpunkt des Objektes

**Modified** Zeitpunkt der letzten Änderung

TimeExternal Datum und Uhrzeit der letzten Änderung der externen Datei vor Import CreatedUser

Benutzer, der das Objekt erzeugt hat ModifiedUser Benutzer, der das Objekt zuletzt geändert hat

**TypeUser** Benutzerdefinierte Typinformation

**TypeMIME** MIME-Typ des Objektes

**SizeDba** Speicherverbrauch des Objektes in der Datenbank

**SizeOrg** Originalgröße des Objektes

**Compression** Kompressionsstufe

## Kontakt

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

## Kontakt

**C16\_BinMove(const vHANDLE aHdl,  
const vHANDLE aDestination) :**  
**vERROR**  
**Binäres Objekt / Verzeichnis verschieben**  
    Deskriptor des Objektes /  
    aHdl  
    Verzeichnisses  
**aDestination** Deskriptor des  
    Zielverzeichnisses  
**Resultat**     vERROR    Fehlerwert  
    Befehle der  
**Siehe**       Programmierschnittstelle,  
                C16\_BinCopy()

Mit dieser Funktion wird das Objekt bzw. Verzeichnis (aHdl) in das Zielverzeichnis (aDestination) verschoben. Das Objekt / Verzeichnis (aHdl) muss dazu exklusiv gesperrt sein (\_BinLock bzw. \_BinSingleLock). Das Wurzelverzeichnis kann nicht verschoben werden.

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

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden Fehlerwerte:

- \_rNoLock    Das Objekt / Verzeichnis ist nicht exklusiv gesperrt.
- \_rExists    Das Objekt / Verzeichnis existiert bereits im Zielverzeichnis (aDestination).
- \_rDeadlock Verklemmung aufgetreten.

## Kontakt

**C16\_BinOpen(const vPHANDLE  
aInstHdl, const vPHANDLE aDirHdl,  
const vCHAR\* aName, vFLAGS aFlags,  
vPHANDLE\* aBinHdl) : vERROR**

Binäres Objekt oder Verzeichnis öffnen  
aInstHdl Instanz-Handle

aDirHdl Deskriptor des  
Elternverzeichnisses  
Objekt- bzw.

**aName** Verzeichnisname

Optionen \_BinLock

Objekt für  
andere

Benutzer  
sperren

Objekt mit

anderen  
Benutzern

sperren

**\_BinSingleLock** Objekt für  
alle

Benutzer

sperren

**\_BinCreate** Objekt

erstellen

**\_BinCreateNew** Objekt  
explizit

erstellen

**\_BinDirectory** Verzeichnis  
öffnen

Deskriptor des binären

**aBinHdl** Objektes

Resultat vERROR Fehlerwert

Befehle der

Siehe Programmierschnittstelle,

[C16\\_BinClose\(\)](#)

Mit dieser Funktion wird ein binäres Objekt bzw. Verzeichnis geöffnet oder neu angelegt.

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben.

In (aDirHdl) wird der Deskriptor des Ausgangsverzeichnisses angegeben.

 Sofern das Ausgangsverzeichnis dem Wurzelverzeichnis entspricht, wird in (aDirHdl) 0 angegeben. Das Wurzelverzeichnis ist immer vorhanden und braucht auch nicht geöffnet zu werden.

## Kontakt

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

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

Folgende Optionen (aFlags) können angegeben werden:

- \_BinCreate Das Objekt wird im Ausgangsverzeichnis erstellt.
- \_BinCreateNew Das Objekt wird explizit im Ausgangsverzeichnis erstellt. Gibt \_rExists zurück, wenn das Objekt schon existiert.
- \_BinLock Das Objekt wird beim Öffnen oder Anlegen für andere Benutzer gesperrt.
- \_BinSharedLock Das Objekt wird beim Öffnen oder Anlegen mit anderen Benutzer gesperrt.
- \_BinSingleLock Das Objekt wird beim Öffnen oder Anlegen für alle Benutzer gesperrt.
- \_BinDirectory Es wird ein Verzeichnis geöffnet.  
 Wird keine Sperroption angegeben, wird das Objekt mit einer gemeinsamen Sperre (\_BinSharedLock) geöffnet.

Die Sperrung eines Objektes bleibt bis zum Schließen des Objektes mit C16\_BinClose(), oder bis sich der Benutzer von der Datenbank abmeldet, erhalten. Änderungen an einem Objekt (Update, Import usw.) können nur bei einer exklusiven Sperre (\_BinLock bzw. \_BinSingleLock) vorgenommen werden.

Der erzeugte Deskriptor wird in (aBinHdl) zurückgegeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden Fehlerwerte:

- \_rLocked Das Objekt bzw. Verzeichnis ist bereits gesperrt.
- \_rNoKey Das Objekt bzw. Verzeichnis existiert nicht. Nächstes Objekt bzw. Verzeichnis gelesen.
- \_rLastRec Das Objekt bzw. Verzeichnis existiert nicht. Letztes Objekt bzw. Verzeichnis gelesen.
- \_rNoRec Das Objekt bzw. Verzeichnis existiert nicht. Kein weiteres Objekt bzw. Verzeichnis vorhanden.
- \_rExists Das Objekt bzw. Verzeichnis existiert bereits.
- \_rNoRights Benutzerrechte nicht ausreichend.
- \_rDeadlock Verklemmung aufgetreten.

C16ERR\_BIN\_OPERATION Versuch ein binäres Objekt im Wurzelverzeichnis anzulegen.

## Kontakt

**C16\_BinReadToFile**(const  
vHANDLE aBinHdl, const vCHAR\*  
aFilename, const vCHAR\*  
aCipherKey) : vERROR  
Binäres Objekt exportieren  
aBinHdl Deskriptor des binären  
aFilename  
Objektes  
Pfad und Dateiname  
der externen Datei  
aCipherKey Verschlüsselungs-Code

Siehe [Befehle der Programmierschnittstelle](#),  
[C16\\_BinReadToMem\(\)](#),  
[C16\\_BinOpen\(\)](#),  
[C16\\_BinWriteFromFile\(\)](#)

Mit dieser Funktion wird der Inhalt des Objektes (aBinHdl) in die externe Datei (aFilename) exportiert. Falls der Objektinhalt verschlüsselt gespeichert wurde, muss in (aCipherKey) der entsprechende Verschlüsselungscode angegeben werden. Bei einem inkorrektten Code ist das Resultat C16ERR\_BIN\_DECRYPTION. Falls das Objekt leer ist, wird C16ERR\_BIN\_NO\_DATA zurückgeliefert.

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

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden Fehlerwerte:

C16ERR_BIN_DECRYPTION	Verschlüsselungs-Code ist falsch.
C16ERR_BIN_NO_DATA	Binäres Objekt ist leer.
C16ERR_BIN_DATA_ERR	Allgemeiner Fehler.
C16ERR_FSI_NO_PATH	Pfad im Namen der externe Datei (aFilename) nicht vorhanden.
C16ERR_FSI_TOO_MANY_OPEN	Maximale Anzahl offener Dateien erreicht.
C16ERR_FSI_ACCESS_DENIED	Zugriff auf externe Datei (aFilename) verweigert.
C16ERR_FSI_INVALID_HANDLE	Datei-Deskriptor von externer Datei (aFilename) ungültig.
C16ERR_FSI_INVALID_DRIVE	Laufwerk im Namen der externen Datei (aFilename) ungültig.
C16ERR_FSI_SHARING_VIOLATION	Zugriffskonflikt bei Zugriff auf externe Datei (aFilename).
C16ERR_FSI_LOCK_VIOLATION	Sperrkonflikt bei Zugriff auf externe Datei (aFilename).
C16ERR_FSI_OPEN_FAILED	Externe Datei (aFilename) konnte nicht geöffnet werden.

## Kontakt

**C16\_BinReadToMem(const vHANDLE aBinHdl, void\* aMemAddress, const vXLONG aMemSize, const vCHAR\* aCipherKey) : vERROR**  
Binäres Objekt exportieren  
aBinHdl Deskriptor des binären  
Objektes  
Adresse des  
**aMemAddress** Zielspeicherbereiches  
aMemSize Größe des  
Speicherbereiches  
**aCipherKey** Verschlüsselungs-Code  
**Resultat** vERROR Fehlerwert  
**Befehle der Programmierschnittstelle,**  
Siehe [C16\\_BinReadToFile\(\)](#),  
[C16\\_BinOpen\(\)](#),  
[C16\\_BinWriteFromMem\(\)](#)

Mit dieser Funktion wird der Inhalt des binären Objekts (aBinHdl) in den Speicherbereich (aMemAddress) eingelesen. Falls der Objektinhalt verschlüsselt gespeichert wurde, muss in (aCipherKey) der entsprechende Verschlüsselungscode angegeben werden. Bei einem inkorrektten Code ist das Resultat C16ERR\_BIN\_DECRYPTION. Falls das Objekt leer ist, wird C16ERR\_BIN\_NO\_DATA zurückgeliefert. In allen anderen Fällen ist das Resultat C16ERR\_OK.



Der Speicherbereich muss mindestens so groß sein, wie das binäre Objekt in entpackter Form (siehe [SizeOrg](#)).

Der Rückgabewert vom Typ [vERROR](#) beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe [Fehlerwerte](#)) oder einen der folgenden Fehlerwerte:

**C16ERR\_BIN\_DECRYPTION** Verschlüsselungs-Code ist falsch.

**C16ERR\_BIN\_NO\_DATA** Binäres Objekt ist leer.

**C16ERR\_BIN\_DATA\_ERR** Allgemeiner Fehler.

Beispiel:

```
// Binäres Objekt öffnen vHANDLE tBinHdl; tErr = C16_BinOpen(tInstHdl, tDirHdl, "Content.txt", 0,
```

## Kontakt

**C16\_BinRename(const vHANDLE aHdl, const  
vCHAR\* aNewName) : vERROR**

**Binäre(s) Datei oder Verzeichnis umbenennen**  
Deskriptor des Objektes /  
aHdl

**Verzeichnisses**

**aNewName** Neuer Name

**Resultat**      vERROR      Fehlerwert

**Befehle der  
Programmierschnittstelle,**

Siehe

**C16\_BinCopy(),**  
**C16\_BinMove()**

Mit dieser Funktion wird das Objekt/Verzeichnis (aHdl) nach (aNewName) umbenannt. Das Objekt/Verzeichnis muss dazu exklusiv gesperrt sein (\_BinLock bzw. \_BinSingleLock).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (**siehe Fehlerwerte**) oder einen der folgenden positiven Fehlerwerte:

**\_rNoLock**      Das Objekt ist nicht exklusiv gesperrt.

**\_rExists**      Das Objekt (aNewName) existiert bereits.

**\_rDeadlock** Verklemmung aufgetreten.

## Kontakt

**C16\_BinUpdate(const vHANDLE aHdl, const vC16\_BinObj\* aBinData) : vERROR**  
Änderungen an binären Objekt / Verzeichnis übernehmen  
aHdl Deskriptor des binären

Objektes / Verzeichnisses

Zu schreibende Daten des

aBinData binären Objektes /  
Verzeichnisses

Resultat vERROR Fehlerwert

Befehle der

Siehe Programmierschnittstelle,  
C16\_BinInfo()

Mit diesem Befehl werden Änderungen an den Eigenschaften eines Objektes oder eines Verzeichnisses in der Datenbank gespeichert. Das Objekt/Verzeichnis muss dazu exklusiv gesperrt sein (\_BinLock bzw. \_BinSingleLock).

Die geänderten Informationen müssen in (aData) in einer Struktur übergeben werden.

Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vXLONG ID; vXLONG StorageID; vCHAR Name[64]; vCHAR FullName[512]; vCHA
```

### Komponente Beschreibung

ID	Identität des Objektes / Verzeichnisses
StorageID	Storage-ID (nur bei binären Objekten)
Name	Name des Objektes / Verzeichnisses
FullName	Pfad und Name des Objektes / Verzeichnisses (nur Verfügbar, wenn das Argument aQueryPath = TRUE ist)
Custom	Benutzerdefinierte Eigenschaft
Created	Erstellungszeitpunkt des Objektes
Modified	Zeitpunkt der letzten Änderung

TimeExternal Datum und Uhrzeit der letzten Änderung der externen Datei vor Import CreatedUser

Benutzer, der das Objekt erzeugt hat ModifiedUser Benutzer, der das Objekt zuletzt geändert hat

TypeUser	Benutzerdefinierte Typinformation
TypeMIME	MIME-Typ des Objektes
SizeDba	Speicherverbrauch des Objektes in der Datenbank
SizeOrg	Originalgröße des Objektes

Compression Kompressionsstufe

Bei binären Verzeichnissen kann nur der Eintrag Custom verändert werden.

Bei binären Objekten können die folgenden Einträge verändert werden:

- Custom
- Created
- Modified
- TimeExternal
- CreatedUser

## Kontakt

- **ModifiedUser**
- **TypeUser**
- **TypeMIME**

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden positiven Fehlerwerte:

**\_rNoLock** Das Objekt / Verzeichnis ist nicht exklusiv gesperrt.

**\_rDeadlock** Verklemmung aufgetreten.

## Kontakt

**C16\_BinWriteFromFile(const vHANDLE aBinHdl, const vCHAR\* aFilename, const vCHAR\* aCipherKey, const vBYTE aCompression) : vERROR**  
**Binäres Objekt importieren**  
    Deskriptor des  
    aBinHdl  
        binären Objektes  
        Pfad und Dateiname  
    aFilename  
        der externen Datei  
    aCipherKey  
        Verschlüsselungs-Code  
    aCompression  
        Kompressionsstufe  
    Resultat  
        vERROR Fehlerwert  
        Befehle der Programmierschnittstelle,  
    Siehe  
        C16\_BinWriteFromMem(),  
        C16\_BinOpen(),  
        C16\_BinReadToFile()

Mit dieser Funktion wird der Inhalt der externen Datei (aFilename) in das Objekt (aBinHdl) importiert. Ein bereits bestehender Inhalt wird dabei überschrieben. Das Objekt muss dazu exklusiv gesperrt sein ( \_BinLock bzw. \_BinSingleLock). Die externe Datei darf nicht leer sein.

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

Der Inhalt kann mit den Stufen 1 bis 4 komprimiert werden. Eine Kompressionsstufe (aCompression) sollte nicht bei Dateien angegeben werden, die sich nicht weiter komprimieren lassen. Dazu gehören vor allem gepackte Dateiformate (.zip, .rar usw.) und komprimierte Multimedia-Formate (.jpg, .mov, .mp3 usw.).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden Fehlerwerte:

_rNoLock	Binäres Objekt (aBinHdl) ist nicht exklusiv gesperrt.
_rDeadlock	Verklemmung aufgetreten.
C16ERR_BIN_NO_DATA	Externe Datei (aFilename) ist leer.
C16ERR_FSI_NO_PATH	Pfad im Namen der externe Datei (aFilename) nicht vorhanden.
C16ERR_FSI_NO_PATH	Datei im Namen der externe Datei (aFilename) nicht vorhanden.
C16ERR_FSI_TOO_MANY_OPEN	Maximale Anzahl offener Dateien erreicht.
C16ERR_FSI_ACCESS_DENIED	Zugriff auf externe Datei (aFilename) verweigert.
C16ERR_FSI_INVALID_HANDLE	Datei-Deskriptor von externer Datei (aFilename)

## Kontakt

<b>C16ERR_FSI_INVALID_DRIVE</b>	ungültig.
<b>C16ERR_FSI_SHARING_VIOLATION</b>	Zugriffskonflikt bei Zugriff auf externe Datei (aFilename) ungültig.
<b>C16ERR_FSI_LOCK_VIOLATION</b>	Sperrkonflikt bei Zugriff auf externe Datei (aFilename).
<b>C16ERR_FSI_OPEN_FAILED</b>	Externe Datei (aFilename) konnte nicht geöffnet werden.

## Kontakt

**C16\_BinWriteFromMem(const vHANDLE aBinHdl, void\* aMemAddress, const vXLONG aMemSize, const vCHAR\* aCipherKey, const vBYTE aCompression) : vERROR**  
Binäres Objekt importieren  
aBinHdl Deskriptor des binären  
aMemAddress Objektes Adresse des  
aMemSize Quellspeicherbereichs Größe des  
aCipherKey Speicherbereichs  
aCompression Verschlüsselungs-Code  
aResultat Kompressionsstufe  
Resultat vERROR Fehlerwert  
Befehle der Programmierschnittstelle,  
Siehe C16\_BinWriteFromFile(),  
C16\_BinOpen(),  
C16\_BinReadToMem()

Mit dieser Funktion wird der komplette Inhalt des Speicherbereichs (aMemAddress / aMemSize) in das binäre Objekt (aBinHdl) importiert. Ein bereits bestehender Inhalt wird dabei überschrieben. Das Objekt muss dazu exklusiv gesperrt sein (\_BinLock bzw. \_BinSingleLock). Der Speicherbereich darf nicht leer sein.

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

Der Inhalt kann mit den Stufen 1 bis 4 komprimiert werden. Eine Kompressionsstufe (aCompression) sollte nicht bei Dateien angegeben werden, die sich nicht weiter komprimieren lassen. Dazu gehören vor allem gepackte Dateiformate (.zip, .rar usw.) und komprimierte Multimedia-Formate (.jpg, .mov, .mp3 usw.).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler), einen negativen Fehlerwert (siehe Fehlerwerte) oder einen der folgenden Fehlerwerte:

\_rNoLock Binäres Objekt (aBinHdl) ist nicht exklusiv gesperrt.  
\_rDeadlock Verklemmung aufgetreten.  
C16ERR\_BIN\_NO\_DATA Der Speicherbereich ist leer.

## Kontakt

Externe Programmierschnittstelle - Wartungs-Funktionen

Übersicht über die Wartungs-Funktionen der Programmierschnittstelle

- C16\_DiagResults
- C16\_DiagStart
- C16\_DiagTerm
- C16\_DiagWatch
- C16\_KeyReorgStart
- C16\_KeyReorgWatch

## Kontakt

**C16\_DiagStart(const vPHANDLE aInstHdl,  
const vLONG aMode) : vERROR**  
**Diagnose / Recover / Schlüsselanalyse starten aInstHdl**  
**Instanz-Handle**  
**aMode Art der Diagnose**  
**Resultat vERROR Fehlerwert**  
**Befehle der**  
**Siehe Programmierschnittstelle,**  
**C16\_DiagWatch()**



Zur Durchführung der Diagnose müssen alle vier Funktionen verwendet werden, um einen definierten Zustand zu behalten. Während der Diagnose können keine anderen Datenbankoperationen der Programmierschnittstelle aufgerufen werden.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Die Art der Diagnose wird in (aMode) übergeben. Folgende Optionen können angegeben werden:

\_DiagExtended Erweiterte Diagnose  
\_DiagRecover mit Recover  
\_DiagKeyReference mit Schlüsselanalyse

Für eine normale Diagnose wird in (aMode) 0 übergeben, die Optionen können kombiniert werden.

Anschließend muss solange C16\_DiagWatch() aufgerufen werden, bis die Variable "Status" den Wert \_DiagMode\_End enthält.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_DiagWatch(const vHANDLE aInstHdl,  
vC16\_DiagInfo\* aInfo) : vERROR**

**Diagnose / Recover / Schlüsselanalyse überwachen aInstHdl**

**Instanz-Handle**

**aInfo      Informationsstruktur**

**Resultat vERROR      Fehlerwert**

**Befehle der  
Programmierschnittstelle,**

Siehe

**C16\_DiagStart(),**  
**C16\_DiagResults()**



Zur Durchführung der Diagnose müssen alle vier Funktionen verwendet werden, um einen definierten Zustand zu behalten. Während der Diagnose können keine anderen Datenbankoperationen der Programmierschnittstelle aufgerufen werden.

Diese Funktion liefert den aktuellen Status der Diagnose.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. In (aInfo) wird ein Zeiger auf eine Informationsstruktur übergeben. Zuvor muss die Strukturkomponente InfoSize mit der Größe der Struktur initialisiert werden (sizeof(vC16\_DiagInfo)).

Die Struktur ist folgendermaßen definiert:

```
typedef struct{  vINT    InfoSize;  vLONG   Status;  vLONG   Progress;}vC16_DiagInfo;
```

### Komponente Beschreibung

**InfoSize**      Größe der Struktur in Byte

**Status**      derzeitiger Status

<u>_DiagMode_None</u>	Die Diagnose wird initialisiert.
<u>_DiagMode_Read1</u>	Strukturdiagnose (1. Lauf)
<u>_DiagMode_Recover</u>	Wiederherstellungslauf
<u>_DiagMode_Read2</u>	Strukturdiagnose (2. Lauf)
<u>_DiagMode_Primes</u>	Primediagnose
<u>_DiagMode_KeyRef</u>	Schlüsselanalyse
<u>_DiagMode_End</u>	Diagnose abgeschlossen

**Progress**      Fortschritt des jeweiligen Laufes in Promille (0-1000)

Die Funktion muss solange aufgerufen werden, bis in der Strukturkomponente Status der Wert \_DiagMode\_End zurückgegeben wird. Anschließend kann das Ergebnis der Diagnose mit der Funktion C16\_DiagResults() ermittelt werden. Der Diagnose-Modus wird mit der Funktion C16\_DiagTerm() beendet.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_DiagResults(const  
vHANDLE aInstHdl,  
vC16\_DiagResult\*\* aResults) :**  
**vERROR**  
Diagnoseergebnis auslesen  
aInstHdl Instanz-Handle  
aResults Zeiger auf  
    Resultattabelle  
    Resultat vERROR Fehlerwert

Siehe

Befehle der  
Programmierschnittstelle,  
C16\_DiagWatch(),  
C16\_DiagTerm()



Zur Durchführung der Diagnose müssen alle vier Funktionen verwendet werden, um einen definierten Zustand zu behalten. Während der Diagnose können keine anderen Datenbankoperationen der Programmierschnittstelle aufgerufen werden.

Diese Funktion gibt einen Zeiger auf das Ergebnis der Diagnose zurück. Zuvor muss der Befehl C16\_DiagWatch() ausgeführt werden und die Diagnose beendet worden sein.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. In (aResults) muss die Adresse eines Zeigers auf eine vC16\_DiagResult-Struktur übergeben werden.

Der Befehl gibt den Zeiger zurück, über den das Diagnoseergebnis ausgelesen werden kann. Das Diagnoseergebnis steht bis zum Aufruf der Funktion C16\_DiagTerm() zur Verfügung. Ein Element der Tabelle enthält folgende Einträge:

```
typedef union{ struct { vBYTE ResultType; vBYTE ResultStatus; vBYTE ErrorCode;
```

**Komponente Beschreibung**

<b>r.ErrorCode</b>	Fehlerwert
<b>r.ResultType</b>	Typ des Resultateintrages
	<u>_DiagRes_End</u> Ende der Tabelle
	<u>_DiagRes_TreeCollision</u> Baumkollision
	<u>_DiagRes_FreeSpace</u> Freie Speicherverwaltung
	<u>_DiagRes_Tree</u> Baumstruktur
	<u>_DiagRes_Prime</u> Prime-Counter
	<u>_DiagRes_File</u> Datei
	<u>_DiagRes_Key</u> Schlüssel
	<u>_DiagRes_Time</u> Dauer der Diagnose

## Kontakt

r.ResultStatus Art des Fehlers (Kombinationen sind möglich)

0	OK (kein Defekt)
_DiagResF_Defect	Defekt
_DiagResF_Repaired	Repariert
_DiagResF_Collisions	Kollision
_DiagResF_RefError	Abweichend
_DiagResF_RefCorrected	Korrigiert
_DiagResF_Incomplete	Unvollständig
_DiagResF_Extensions	weitere Informationen in der nächsten Zeile

r.KeyNo

Schlüsselnummer, wenn r.ResultType = \_DiagRes\_Key

r.TreeNo

Baumnummer, wenn r.ResultType = \_DiagRes\_Tree oder  
\_DiagRes\_Prime

r.FileNo

Dateinummer, wenn r.ResultType = \_DiagRes\_File oder \_DiagRes\_Key

r.Counter

Anzahl der Einträge

Ist in der Komponente ResultStatus der Wert \_DiagResF\_Extensions gesetzt, stehen erweiterte Informationen im nächsten Eintrag der Tabelle:

*r.ResultType = \_DiagRes\_File*

Komponente Beschreibung

x.ExtInfo1	Anzahl der unvollständigen Sätze
x.ExtInfo2	Anzahl der gelöschten defekten Sätze
x.ExtInfo3	Anzahl der gelöschten Datensatzfragmente

*r.ResultType = \_DiagRes\_Key*

Komponente Beschreibung

x.ExtInfo1	Anzahl der eingefügten Schlüssel
x.ExtInfo2	Anzahl der gelöschten Schlüssel
x.ExtInfo3	Anzahl der Wertekollisionen

Nach der Auswertung des Ergebnisses muss die Funktion [C16\\_DiagTerm\(\)](#) aufgerufen werden.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

Der Rückgabewert kann mit folgenden Konstanten verglichen werden:

C16ERR_OK	Datenbank "OK"
C16ERR_DIAG_DEFECT	Datenbank "DEFEKT"
C16ERR_DIAG_REPAIRED	Datenbank "REPARIERT"
C16ERR_DIAG_CORRECTED	Datenbank "KORRIGIERT"
C16ERR_DIAG_MEMORY	Haupspeicher des Servers nicht ausreichend
C16ERR_DIAG_READ_ONLY	Datenbank ist Read-Only (nur bei Recover-Option)
C16ERR_DIAG_SEQUENCE	Aufruf vor dem Ende des Diagnoselaufes
C16ERR_DIAG_OTHER	sonstiger Fehler

## Kontakt

**C16\_DiagTerm(const  
vHANDLE aInstHdl) : vERROR**  
Diagnosemodus beenden  
aInstHdl Instanz-Handle  
Resultat vERROR Fehlerwert

Siehe Befehle der Programmierschnittstelle,  
C16\_DiagResults()

 Zur Durchführung der Diagnose müssen alle vier Funktionen verwendet werden, um einen definierten Zustand zu behalten. Während der Diagnose können keine anderen Datenbankoperationen der Programmierschnittstelle aufgerufen werden.

Diese Funktion beendet den Diagnose-Modus. Zuvor muss das Ergebnis der Diagnose mit dem Befehl C16\_DiagResults() ausgewertet worden sein.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_KeyReorgStart(const  
vHANDLE aInstHdl,  
const vINT aFileNo,  
const vINT aKeyNo) : vERROR**  
Schlüsselreorganisation starten  
aInstHdl Instanz-Handle  
aFileNo Dateinummer  
aKeyNo Schlüsselnummer  
Resultat vERROR Fehlerwert

Befehle der

Siehe     Programmierschnittstelle,  
C16\_KeyReorgWatch()

Mit dieser Funktion wird die Reorganisation des Schlüssels (aKeyNo) in der Datei (aFileNo) gestartet. Die Funktion kehrt sofort zurück und liefert C16ERR\_OK, wenn der Start erfolgreich war. Bei ungültigen Datei- oder Schlüsselnummern wird C16ERR\_NO\_FILE oder C16ERR\_NO\_KEY zurückgeliefert. Konnte der Server die Datenstruktur nicht laden oder den Reorganisationsthread nicht starten, wird der Fehler C16ERR\_LOAD\_DS bzw. C16ERR\_SERVER\_ASYNC (siehe Fehlerwerte) zurückgegeben.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Nach dem Starten der Schlüsselreorganisation muss so lange der Befehl C16\_KeyReorgWatch() aufgerufen werden, bis die Variable "Result" den Wert C16RES\_ASYNC\_TERM oder C16RES\_ASYNC\_BREAK enthält. Bis zu diesem Zeitpunkt können keine anderen Datenbankoperationen der DLL aufgerufen werden.

## Kontakt

```
C16_KeyReorgWatch(const  
vHANDLE aInstHdl,  
const vBOOL aAbort,  
vC16_KeyReorgInfo* aInfo) :  
vERROR  
Schlüsselreorganisation überwachen  
aInstHdl Instanz-Handle  
aAbort Reorganisation abbrechen  
Informationen zur  
aInfo
```

### Reorganisation

Resultat vERROR Fehlerwert

#### Befehle der

Siehe Programmierschnittstelle,  
C16\_KeyReorgStart()

Diese Funktion liefert den aktuellen Status des Reorganisationslaufes. Die Informationen befinden sich nach dem Aufruf in einer Struktur vom Typ vC16\_KeyReorgInfo. Die Struktur ist folgendermaßen definiert:

```
typedef struct{ vINT InfoSize; vXLONG RecID; vLONG KeyCount; vLONG RecErrorCount; vLONG
```

Komponente	Beschreibung
InfoSize	Größe der Struktur in Byte
RecID	<u>Datensatz-ID des gerade bearbeiteten Datensatzes</u>
KeyCount	Anzahl der bisher generierten Schlüsseleinträge
RecErrorCount	Anzahl der bisher entdeckten defekten Datensätze
KeyErrorCount	Anzahl der bisher aufgetretenen Wertekollisionen
Result	Ergebnis der Reorganistation
	C16RES_ASYNC_RUNNING 1      Schlüsselreorganisation wird gerade durchgeführt
	C16RES_ASYNC_TERM            2 Schlüsselreorganisation vollständig durchgeführt
	C16RES_ASYNC_BREAK          3 Schlüsselreorganisation wurde abgebrochen

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Über den Parameter (aAbort) kann die Schlüsselreorganisation abgebrochen (aAbort = true) werden. Beim Abbruch der Schlüsselreorganisation kann es zu einer Verzögerung von 1-2 Sekunden kommen. In (aInfo) muss ein Zeiger auf eine vC16\_KeyReorgInfo-Struktur übergeben werden. Die Strukturkomponente InfoSize muss vor der Übergabe mit der Größe der Struktur initialisiert werden (sizeof(vC16\_KeyReorgInfo)).

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

Die Funktion muss solange aufgerufen werden, bis in der Strukturkomponente Result C16RES\_ASYNC\_BREAK oder C16RES\_ASYNC\_TERM zurückgegeben wird.

## Kontakt

Externe Programmierschnittstelle - Sonstige Funktionen

Übersicht über die sonstigen Funktionen der Programmierschnittstelle

- [C16\\_ProcArgument](#)
- [C16\\_ProcCall](#)
- [C16\\_ProcCompile](#)
- [C16\\_ProcOpen](#)
- [C16\\_ProcResult](#)
- [C16\\_ServerBackup](#)
- [C16\\_SetCharDefs](#)
- [C16\\_UserClear](#)

**C16\_ProcArgument(const vHANDLE aInstHdl, const vBYTE aArgType, void\* aArgVal, const vBOOL aVarArg, const vBYTE aMaxLen) :**  
vERROR  
Prozedurargument übergeben  
aInstHdl Instanz-Handle  
aArgType Typ des Arguments  
aArgVal Wert des Arguments  
Übergabe als Referenz (true),  
aVarArg  
Übergabe als Wert (false)  
Maximale Länge des  
aMaxLen zurückgegebenen  
Alphawertes  
Resultat vERROR Fehlerwert

Befehle der Programmierschnittstelle,  
Siehe  
[C16\\_ProcCall\(\)](#),  
[C16\\_ProcResult\(\)](#)

Mit dieser Funktion können vor dem Aufruf einer Prozedur mit [C16\\_ProcCall\(\)](#) Argumente definiert werden. Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der in der Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben.  
(aArgType) beinhaltet den Typ des Übergabeparameters. Zur Festlegung des Types können folgende Konstanten verwendet werden:

aArgType	Parametertyp
_TypeAlpha	vCHAR (C++) / vPCHAR (Delphi)
_TypeByte	vBYTE
_TypeCaltime	vCALTIME
_TypeDate	vDATE
_TypeFloat	vFLOAT
_TypeHandle	vHANDLE
_TypeInt	vINT
_TypeLogic	vBOOL
_TypePoint	vPOINT
_TypeRect	vRECT
_TypeTime	vTIME
_TypeWord	vWORD

In (aArgVal) wird der Wert des Parameters übergeben. Die Übergabe kann als Wert (aVarArg = false) oder als Referenz (aVarArg = true) erfolgen. (aMaxLength) wird nur benötigt, wenn in (aArgVal) eine Zeichenkette als VAR-Parameter (aVarArg = true) übergeben wird. Der Wert gibt die maximale Anzahl der Zeichen an, die

## Kontakt

zurückgegeben werden können. In allen anderen Fällen wird der Wert ignoriert.

 Sollen einer Prozedur mehrere Argumente übergeben werden, muss die Funktion für jedes Argument aufgerufen werden.

 Bei der Übergabe als Referenz (`aVarArg = true`) ist zu beachten, dass das entsprechende Argument der durch C16\_ProcCall aufzurufenden CONZEPT 16-Funktion nicht mit var definiert sein darf.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

Beispiel:

```
Deklaration der CONZEPT 16-Funktion: sub C16_Proc( aText : alpha; aValue : int; ): logic; DLL-Sch
```

**C16\_ProcCall(const vPHANDLE  
aInstHdl, const vCHAR\*  
aProcName) : vERROR**  
**CONZEPT 16-Funktion aufrufen**  
aInstHdl Instanz-Handle  
aProcName Prozedurname in  
der Datenbank  
Resultat vERROR Fehlerwert  
Befehle der  
Programmierschnittstelle,  
Siehe C16\_ProcArgument(),  
C16\_ProcResult()

Mit dieser Funktion können Prozeduren, die in der Datenbank erstellt wurden, ausgeführt werden. Die Prozedur muss mit einem CONZEPT 16-Client ab Version 4.2 erstellt worden sein. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben. Bei C16\_OpenArea() muss ein Prozedurcache größer 0 eingetragen worden sein. Der Prozedurname wird im Parameter (aProcName) übergeben.



Von der Programmierschnittstelle können nur Prozeduren ausgeführt werden, die mit der Option A+ übersetzt wurden. Alle Befehle, die in diesen Prozeduren verwendet werden können, sind in der Dokumentation mit dem Symbol versehen. Wird die Programmierschnittstelle mit grafischer Erweiterung verwendet, können zusätzlich Befehle mit dem Symbol verwendet werden.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_ProcCompile(const vHANDLE aInstHdl, vCHAR\* aProcName, const vCHAR\* aLangID, vC16\_CompiledInfo\* aInfoBlock) : vERROR**  
Prozedur übersetzen  
aInstHdl Instanz-Handle  
Name der zu  
aProcName übersetzenden  
Prozedur  
Sprach-ID oder  
aLangID NULL  
Informationsstruktur  
aInfoBlock oder NULL  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle,  
C16\_ProcCall()

Mit dieser Anweisung wird die in (aProcName) übergebene Prozedur übersetzt. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Mit dem Argument (aLangID) kann die Sprache für Fehlermeldungen definiert werden.

Folgende Werte können abgegeben werden:

"DE" Deutsch  
"EN" / NULL U.S. Englisch  
"\*U" Systemsprache des aktuellen Windows-Benutzers

Bei Fehlern wird, sofern übergeben, eine Struktur vom Typ vC16\_CompiledInfo gefüllt.

Die Struktur hat den folgenden Aufbau:

```
typedef struct{ vERROR ErrorCode;  vINT   ErrorLine;  vINT   ErrorColumn;  vCHAR  ErrorProcSource
```



Die Struktur sollte zuvor mit memset geleert werden.

Komponente	Beschreibung
ErrorCode	Fehlerwert (Identisch mit Rückgabewert)
ErrorLine	Nummer der Fehlerzeile im Quelltext
ErrorColumn	Fehlerposition in der Zeile
ErrorProcSource	Name der Quelltextprozedur (kann vom Namen der übersetzten Prozedur abweichen, wenn eine Prozedur mit der <u>Include-Anweisung</u> eingebunden wird)

ErrorLineText Inhalt der Quelltextzeile in der der Fehler aufgetreten ist ErrorCodeText  
Fehlertext



Unter folgenden Umständen kann eine Prozedur nicht übersetzt werden:

## Kontakt

- Die Prozedur enthält ein with-Konstrukt.
- Die Prozedur wird gerade ausgeführt oder ist von einem anderen Client gesperrt.
- Die Prozedur ist keine A+ Prozedur (siehe [@A+](#)).

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)). Ist der Rückgabewert gleich **C16ERR\_INSTANCE\_HDL\_INVALID**, ist der übergebene (aInstHdl) ungültig.

Fehlercodes zwischen -1104 und -1154 entsprechen den Fehlercodes zwischen -104 und -154, die unter [Fehlermeldungen beim Übersetzen mit dem Editor](#) beschrieben sind.

**Beispiel:**

```
memset(&tCompileInfo, 0, sizeof(tCompileInfo)); tErr = C16_ProcCompile(tInstHdl, tProcName, "DE",
```

**C16\_ProcOpen(const  
vHANDLE aInstHdl,  
vHANDLE\* aProcHdl) :**  
**vERROR**  
Textpuffer für Prozedur anlegen  
aInstHdl Instanz-Handle  
aProcHdl Text-Handle auf  
Prozedurpuffer  
Resultat vERROR Fehlerwert

Siehe

Befehle der  
Programmierschnittstelle,  
C16\_TextClose(),  
C16\_TextReadData()

Der Befehl erzeugt einen Textpuffer für die angegebene Prozedur und stellt den dazugehörigen Text-Handle in (aProcHdl). Ein Textpuffer wird für alle weiteren Textoperationen benötigt. Es können beliebig viele Textpuffer angelegt werden. Wird der Textpuffer nicht mehr verwendet, so muss er mit C16\_TextClose() wieder entfernt werden.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der in der Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte). Ist der Rückgabewert gleich **C16ERR\_INSTANCE\_HDL\_INVALID**, ist der übergebene (aInstHdl) ungültig.

## Kontakt

**C16\_ProcResult(const  
vHANDLE aInstHdl,  
vBYTE\* aResultType,  
void\*\* aResultData) : vERROR**  
Prozedurergebnis ermitteln  
**aInstHdl** Instanz-Handle  
**aResultType** Typ des Ergebnisses  
Referenz auf das  
**aResultData** Ergebnis der  
Prozedur  
**Resultat** vERROR Fehlerwert  
Befehle der  
**Siehe** Programmierschnittstelle,  
C16\_ProcCall()

Mit dieser Funktion kann der Ergebniswert einer Prozedur ermittelt werden. Die Prozedur muss zuvor mit C16\_ProcCall() aufgerufen worden sein. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

### Beispiel

```
vCHAR tParameter[11] = "Text";vINT tParameter = 5;vBYTE tType;vBOOL* tResult = NULL;err = C16
```

## Kontakt

**C16\_ServerBackup(const vHANDLE  
aInstHdl,  
const vINT aMinutes) : vERROR**  
Datenbank in Backup-Modus versetzen  
aInstHdl Instanz-Handle  
aMinutes Dauer des Backupstatus  
Resultat vERROR Fehlerwert  
Befehle der  
Siehe Programmierschnittstelle

Diese Funktion startet ein Sicherungssereignis des Servers für die durch (aInstHdl) bezeichnete Datenbank. (aInstHdl) ist der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank übergeben wurde. In (aMinutes) wird die Dauer des Sicherungssereignisses in Minuten angegeben. Während der Sicherung werden nur Lesezugriffe auf die Datenbank zugelassen. Schreibzugriffe werden in die Transaktionsdatei aufgenommen.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_SetCharDefs(const vPHANDLE  
aInstHdl,  
vCHAR aDecimalSep,  
vCHAR aThousandsSep,  
vCHAR aDateSep,  
vCHAR aTimeSep1,  
vCHAR aTimeSep2,  
vCHAR aCharTRUE,  
vCHAR aCharFALSE,  
vCHAR a CharSetConversion) : vERROR**

Zeichenformat festlegen

aInstHdl	Instanz-Handle
aDecimalSep	Dezimaltrennzeichen
aThousandsSep	Tausendertrennzeichen
aDateSep	Datumstrennzeichen
aTimeSep1	Trennzeichen
aTimeSep2	Trennzeichen für hundertstel Sekunden
aCharTRUE	Zeichen für logisch WAHR
aCharFALSE	Zeichen für logisch FALSCH
a CharSetConversion	Zeichenkonvertierung
Resultat	vERROR Fehlerwert
Siehe	<u>Befehle der Programmierschnittstelle,</u> <u>C16_FldDataChar()</u> , <u>C16_RegExtFld()</u>

Diese Funktion legt das Zeichenformat für die ASCII-Konvertierung der Feldinhalte fest. Die gewählten Zeichenformate gelten nur für die jeweilige Datenbank-Instanz. Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

In den Parametern (aDecimalSep) und (aThousandsSep) werden die Trennzeichen für Zahlen angegeben. Für das Dezimaltrennzeichen ist "." und für das Tausendertrennzeichen "," voreingestellt.

Das Trennzeichen für die Kurzform des Datums wird im Parameter (aDateSep) angegeben. Voreingestellt ist hier ":".

Die Trennzeichen für Zeitwerte unterteilen sich in das Trennzeichen zwischen Stunden/Minuten/Sekunden (aTimeSep1) und Sekunden/Hundertstelsekunden (aTimeSep2). Voreingestellt sind ":" und ":".

Die Zeichen für Wahrheitswerte werden in den Parametern (aCharTRUE) und (aCharFALSE) angegeben. Voreingestellt ist "Y" für wahr und "N" für falsch.

In (a CharSetConversion) wird eine symbolische Konstante angegeben, welche die Zeichenkonvertierung angibt, die bei der Übertragung von alphanumerischen

## Kontakt

Feldinhalten vorgenommen werden soll. Folgende Konstanten sind definiert:

**\_FldExc\_CharSet\_C16** Alphanumerische Feldinhalte werden mit einem CONZEPT 16 eigenen, internen Zeichensatz ausgegeben. Es findet somit keine Zeichensatzkonvertierung statt (dies ist die voreingestellte Zeichenkonvertierung).

**\_FldExc\_CharSet\_OEM** Alphanumerische Feldinhalte werden in den entsprechenden OEM-Zeichensatz konvertiert

**\_FldExc\_CharSet\_ANSI** Alphanumerische Feldinhalte werden in den ANSI-Zeichensatz konvertiert

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_UserClear(const vPHANDLE aInstHdl,  
const vINT aUserID,  
const vINT aUserNo,  
const vCHAR\* aServerPassword) :**  
**vERROR**  
Benutzer aus der Datenbank entfernen  
aInstHdl            Instanz-Handle  
aUserId            Benutzer-ID  
aUserNo            Benutzernummer  
                    Kennwort des  
aServerPassword Servers  
Resultat            vERROR Fehlerwert  
                    Befehle der  
Siehe              Programmierschnittstelle,  
                    C16\_QueryUserInfo()

Im Parameter (aInstHdl) wird der von [C16\\_InitInstance\(\)](#) bereitgestellte Instanz-Handle, der an die Funktion [C16\\_OpenArea\(\)](#) zum Öffnen der Datenbank verwendet wurde, übergeben. In (aUserID) und (aUserNo) werden die Benutzer-ID und die Benutzernummer des zu entfernenden Benutzers übergeben. Die entsprechenden Parameter können mit dem Befehl [C16\\_QueryUserInfo\(\)](#) ermittelt werden. In (aServerPassword) wird das Kennwort des Servers oder ein Leerstring übergeben.

Die Benutzer-ID und die Benutzernummer müssen zum selben Benutzer gehören, damit er ausgeloggt werden kann.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe [Fehlerwerte](#)).

### Beispiel

```
vCHAR* tSrvPwd = "";vERROR error;error = C16_UserClear(instHdl,UserInfo.ID,UserInfo.Number,tSrvPwd
```

## Kontakt

Externe Programmierschnittstelle - Transaktions-Funktionen

Übersicht über die Transaktions-Funktionen der Programmierschnittstelle

- C16\_DtaBegin
- C16\_DtaCommit
- C16\_DtaRollback

## Kontakt

**C16\_DtaBegin(const vHANDLE**

**aInstHdl) : vERROR**

**Transaktion starten**

**aInstHdl Instanz-Handle**

**Resultat vERROR Fehlerwert**

**Siehe** Befehle der Programmierschnittstelle,

**C16\_DtaCommit(),**  
**C16\_DtaRollback()**

**Ab dem Aufruf dieser Systemfunktion ist eine Transaktion aktiviert, alle nachfolgenden Datenoperationen werden als Einheit betrachtet, bis C16\_DtaCommit() oder C16\_DtaRollback() erfolgt. Innerhalb einer Transaktion können weitere Transaktionen gestartet werden, sofern ein CONZEPT 16-Datenbankserver benutzt wird.**

**Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.**

**Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).**

**Beispiel:**

```
#include "c16_pgif.h"extern vHANDLE ghInstanz;BOOL consistChange(void *pData){    int nErg; // Fun
```

## Kontakt

**C16\_DtaCommit(const  
vHANDLE aInstHdl) : vERROR**  
Transaktion beenden  
aInstHdl Instanz-Handle  
Resultat vERROR Fehlerwert

Siehe  
Befehle der  
Programmierschnittstelle,  
C16\_DtaBegin(),  
C16\_DtaRollback()

Diese Funktion beendet eine aktive Transaktion. Alle während der Transaktion vorgenommenen Datenänderungen werden dann endgültig gespeichert. Im Parameter (aInstHdl) wird der von **C16\_InitInstance()** bereitgestellte Instanz-Handle, der an die Funktion **C16\_OpenArea()** zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ **vERROR** beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

**C16\_DtaRollback(const vHANDLE aInstHdl, const vBOOL aAllLevels) : vERROR**  
Transaktion abbrechen  
aInstHdl Instanz-Handle  
alle Transaktionen  
aAllLevels (true), letzte  
Transaktion (false)  
abbrechen  
**Resultat vERROR Fehlerwert**  
**Befehle der Programmierschnittstelle,**  
**Siehe**  
**C16\_DtaBegin(),**  
**C16\_DtaCommit()**

Diese Funktion bricht eine aktive Transaktion ab. Alle während der Transaktion vorgenommenen Datenänderungen werden dadurch rückgängig gemacht, und es ist wieder der Zustand bei Transaktionsanfang hergestellt. Bei geschachtelten Transaktionen kann durch (aAllLevels) angegeben werden, ob alle offenen Transaktionen (true) oder nur die letzte Transaktion (false) zurückgesetzt werden sollen.

Im Parameter (aInstHdl) wird der von C16\_InitInstance() bereitgestellte Instanz-Handle, der an die Funktion C16\_OpenArea() zum Öffnen der Datenbank verwendet wurde, übergeben.

Der Rückgabewert vom Typ vERROR beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

## Kontakt

### Datenbankzugriffe

Beispiele zur Programmierung mit der Programmierschnittstelle

In diesem Abschnitt wird in einigen Beispielen die Programmierung mit der Programmierschnittstelle vorgestellt. Folgende Beispiele werden erläutert:

- Öffnen und Schließen einer Datenbank
- Abfragen von Informationen
- Datensatzoperationen
- Datentransfer
- Automatischer Datentransfer

## Kontakt

### Öffnen und Schließen einer Datenbank

Beispiel zum Öffnen und Schließen einer Datenbank mit der Programmierschnittstelle Um auf eine Datenbank zuzugreifen, muss zuerst die DLL initialisiert werden. Anschließend wird eine Datenbank-Instanz generiert, mit der die Datenbank geöffnet werden kann.

1. Zuerst wird C16\_InitPgif() aufgerufen, um die DLL zu initialisieren und Hauptspeicher für das Laden der Datenstrukturen zu reservieren. Diese Funktion muss nur einmal zu Programmbeginn aufgerufen werden.
2. Für jede Datenbank, auf die zugegriffen wird, muss eine eigene Datenbank-Instanz initialisiert werden. Dies erfolgt durch die Übergabe des von C16\_InitPgif() bereitgestellten Modul-Handles an die Funktion C16\_InitInstance().
3. Schließlich wird der von C16\_InitInstance() bereitgestellte Instanz-Handle an C16\_OpenArea() übergeben, um die Datenbank zu öffnen und deren Datenstruktur zu laden.

Danach kann auf die Datenbank und die Datenstruktur zugegriffen werden. Vor Beendigung des Programms müssen alle geöffneten Datenbanken geschlossen und die Instanzen beendet werden.

### Beispiel

```
#include "c16_pgx.h"extern void MsgBox(char *lpszMsg);static vCHAR *GetUserName();static vCHAR *G
```

Im obigen Beispiel werden die DLL und eine Datenbank-Instanz initialisiert. Anschließend wird die Datenbank T40.CA1, die sich auf dem Server CENTRAL befindet, geöffnet. Nach Verarbeitung der Daten wird die Datenbank geschlossen und die Instanz sowie das Modul beendet.

## Kontakt

### Abfragen von Informationen

Beispiel zur Abfrage von Informationen mit der Programmierschnittstelle Öffnen und Schließen

Siehe einer Datenbank,  
Datensatzoperationen

Die CONZEPT 16-Programmierschnittstelle stellt dem Programmierer eine Vielzahl von Funktionen zur Verfügung, um jede Art von Informationen über die Elemente einer Datenbank zu ermitteln: Dateien, Teildatensätze, Felder, Schlüssel, Verknüpfungen usw. Bei den meisten Elementen gibt es jeweils zwei Funktionen zur Ermittlung allgemeiner Informationen - über die Nummer des Elementes und über dessen Namen.

Die Info-Befehle haben alle die gleiche Funktionsweise: Der entsprechenden Funktion werden der jeweilige Instanz-Handle und elementspezifische Angaben zusammen mit einem Zeiger auf eine Info-Struktur, in welche die Informationen geschrieben wird übergeben (die Strukturkomponente InfoSize muss vor der Übergabe initialisiert werden). Als Ergebnis liefern die Informationsfunktionen einen Wert vom Typ vERROR zurück. Ist dieser Wert gleich 0, kann die gewünschte Information aus der Info-Struktur gelesen werden, andernfalls ist ein Fehler aufgetreten.

### Beispiel

```
#include #include "c16_pgx.h"extern void MsgBox(char *lpszMsg); // Instanz-Handleextern vHANDLE g
```

Die Anzahl der Teildatensätze in Datei nFileNo wird ermittelt. Gibt es die Datei nicht oder tritt ein Verarbeitungsfehler auf, wird eine entsprechende Meldung ausgegeben und 0 zurückgegeben.

## Kontakt

### Datensatzoperationen

Beispiel zur Programmierung von Datensatzoperationen mit der Programmierschnittstelle

Datentransfer,  
Siehe Abfragen von  
Informationen

Außer dem vollständigen Zugriff auf die Datenstrukturinformationen stellt die CONZEPT 16-Programmierschnittstelle alle wesentlichen Funktionen für das Datenhandling zur Verfügung. Die Art und Handhabung der Funktionen sind eng an die CONZEPT 16-Prozedurbefehle angelehnt. Beim Arbeiten mit mehreren geöffneten Datenbanken wird die Auswahl durch Angabe des Handles für die jeweilige Instanz vorgenommen. Der Zugriff auf die Feldinhalte kann neben dem binären Datenformat auch mittels ASCII-Werten erfolgen.

### Beispiel

```
#include "c16_pgx.h"extern vHANDLE ghInstanz;BOOL UpdateLast(void *pData) { // Funktionsresultat
```

In diesem Beispiel wird das erste Feld des letzten Datensatzes in Datei 100 geändert und zurückgeschrieben. Tritt kein Fehler während der Verarbeitung auf, wird true zurückgegeben, sonst false.

## Kontakt

### Datentransfer

#### Übertragung der Feldinhalte zwischen den Feldpuffern und den Variablen

Siehe Datensatzoperationen,

Siehe Automatischer

Datentransfer

Mit den Befehlen C16\_RecRead() und C16\_RecLink() können die Datensätze in der Datenbank gelesen werden. Die Informationen stehen zunächst nur in den Feldpuffern. Der Zugriff auf die Feldinhalte kann neben dem binären Datenformat (z. B. mit C16\_FldData()) auch mittels ASCII-Werten erfolgen (z. B. C16\_FldDataChar()). In diesem Fall erfolgt die Konvertierung gemäß den voreingestellten Zeichenformaten (siehe C16\_SetCharDefs()). Wird eine andere Formatierung gewünscht, kann man sie mit C16\_SetCharDefs() festlegen.

## Kontakt

### Automatischer Datentransfer

Übertragung aller Feldinhalte zwischen den Feldpuffern und den Variablen Siehe  
Datentransfer,

#### Datenbankzugriffe

Oft will der Anwendungsentwickler stets auf die gleichen Felder im Feldpuffer der Datei zugreifen, um anschließend deren Inhalte mit den Feldinhalten von in seinem Programm definierten Variablen auszutauschen. Mit den nachfolgenden Funktionen bietet die Programmierschnittstelle eine effiziente Möglichkeit, den Datenaustausch mittels einer direkten Verbindung zwischen Feld und Variable zu automatisieren. Dies erspart den wiederholten Aufruf der Funktion C16\_FldData() bzw. C16\_FldDataChar(), die bei jedem Aufruf die Feldangaben überprüft und beschleunigt somit den Datentransfer erheblich, da die Überprüfung bereits bei der Erstellung der Verbindung erfolgt.

Darüber hinaus kann mit einem einmaligen Aufruf von C16\_SetExcMode() der automatische Hin- und Rücktransfer von Daten zwischen allen Feldern im Feldpuffer der Datei und den jeweils zu ihnen registrierten externen Feldern veranlasst werden. Auf diese Weise werden die Programmvariablen automatisch nach jeder Datentransfer-Operation aktualisiert, z. B. C16\_RecRead().

## Kontakt

Fehlerwerte (externe Windows Programmierschnittstelle)

Rückgabewerte der Funktionen der Programmierschnittstelle

### Externe Windows

Siehe Programmierschnittstelle,

### Datenbankzugriffe

Alle Funktionen der CONZEPT 16-Programmierschnittstelle liefern ein Ergebnis vom Typ vERROR (signed int) zurück. Bei einem Rückgabewert = 0 wurde die Funktion erfolgreich ausgeführt. Im Falle eines negativen Wertes ist ein Fehler aufgetreten, der anhand des genauen Rückgabewertes Informationen zur Fehlerursache liefert. Diese sind in der nachfolgenden Übersicht zusammengefasst:

Code	Symbolische Konstante	Kurzbeschreibung
0	C16ERR_OK	Operation erfolgreich ausgeführt.
-1	C16ERR_GENERIC	Ein allgemeiner Fehler ist aufgetreten.
-2	C16ERR_TIMEOUT	Es ist eine Zeitüberschreitung aufgetreten.
-12	C16ERR_MEM_EXHAUSTED	Hauptspeicher nicht ausreichend.
-13	C16ERR_MEM_DAMAGED	Hauptspeicherverwaltung defekt.
-20	C16ERR_FSI_NO_FILE	Externe Datei oder Verzeichnis nicht vorhanden.
-21	C16ERR_FSI_NO_PATH	Pfad nicht vorhanden.
-22	C16ERR_FSI_TOO_MANY_OPEN	Zuviele offene Dateien.
-23	C16ERR_FSI_ACCESS_DENIED	Zugriff verweigert.
-24	C16ERR_FSI_INVALID_HANDLE	Ungültiger Dateideskriptor.
-25	C16ERR_FSI_INVALID_DRIVE	Ungültige Laufwerksangabe.
-26	C16ERR_FSI_CURRENT_DIR	Aktuelles Verzeichnis kann nicht gelöscht werden.
-27	C16ERR_FSI_SHARING_VIOLATION	Zugriffskonflikt bei externer Datei.
-28	C16ERR_FSI_LOCK_VIOLATION	Sperrkonflikt in externer Datei.
-29	C16ERR_FSI_OPEN_FAILED	Externe Datei konnte nicht geöffnet oder angelegt werden.
-31	C16ERR_FSI_READFAULT	Fehler beim Lesen einer externen Datei.
-32	C16ERR_FSI_WRITEFAULT	Fehler beim Schreiben einer externen Datei.
-39	C16ERR_FSI_OTHER	Unbekannter Fehler beim Arbeiten mit externen Dateien.
-201	C16ERR_INIT_FAILED	Die DLL konnte nicht

## Kontakt

-202	<b>C16ERR_MODULE_HDL_INVALID</b>	initialisiert werden.
-203	<b>C16ERR_INSTANCE_HDL_INVALID</b>	Der Modul-Handle ist ungültig. Entweder der Instanz-Handle ist ungültig, oder die Datenbank wurde nicht geöffnet.
-204	<b>C16ERR_BUFFER_HDL_INVALID</b>	Der Puffer-Handle ist ungültig.
-205	<b>C16ERR_FUNCTION_NOT_AVAILABLE</b>	Funktion ist in der Schnittstellen-Version nicht verfügbar.
-206	<b>C16ERR_NO_MULTIPLE_INSTANCES</b>	Es können nicht mehrere Datenbanken gleichzeitig geöffnet werden.
-207	<b>C16ERR_TEXT_HDL_INVALID</b> -208 <b>C16ERR_SEL_HDL_INVALID</b> -209 <b>C16ERR_BIN_HDL_INVALID</b>	Text-Handle ungültig. Selektions-Handle ungültig Deskriptor eines binären Objektes bzw. Verzeichnisses ungültig
-301	<b>C16ERR_OUT_OF_MEMORY</b>	Nicht genügend Hauptspeicher.
-302	<b>C16ERR_COMM_FAILED</b>	Bei einer Datenbankfunktion ist ein Abbruch der Verbindung zum CONZEPT 16-Server aufgetreten. Bei der Verarbeitung durch den Server ist ein Fehler aufgetreten.
-303	<b>C16ERR_SERVER_OP</b>	Der Server konnte den Reorganisationsthread nicht starten.
-304	<b>C16ERR_SERVER_ASYNC</b>	Serververbindung nicht zustande gekommen.
-401	<b>C16ERR_NO_SERVER_CONNECTION</b>	Die Datenbank konnte auf dem angegebenen Server nicht gefunden werden.
-402	<b>C16ERR_AREA_NOT_FOUND</b>	Allgemeiner Fehler beim Öffnen der Datenbank. Z. B. ist die Datenbank nicht kompatibel mit dem CONZEPT 16-Server oder der Programmierschnittstelle.
-403	<b>C16ERR_AREA_OPEN</b>	Die Datenbank ist gesperrt.
-404	<b>C16ERR_AREA_LOCKED</b> -405 <b>C16ERR_AREA_IN_USE</b>	Die Datenbank ist von dem Server in exklusiver Benutzung.

## Kontakt

-406	<b>C16ERR_AREA_TYPE</b>	Die Datenbank ist nicht kompatibel mit der CONZEPT 16-Version 3.5 oder höher.
-407	<b>C16ERR_AREA_PASSWORD</b>	Das zum Öffnen der Datenbank angegebene Passwort ist inkorrekt.
-408	<b>C16ERR_SERVER_USER_LIMIT</b>	Die maximale Anzahl der zugelassenen Benutzer wurde erreicht.
-409	<b>C16ERR_SERVER_START</b>	Es konnte kein Server-Prozess gestartet werden (z. B. weil der Server gerade heruntergefahren wird, ggf. ein paar Sekunden warten und den Versuch wiederholen).
-410	<b>C16ERR_USER_PASSWORD</b>	Entweder der Benutzer ist unbekannt, oder das Passwort ist ungültig.
-411	<b>C16ERR_DATA_STRUCTURE</b>	Keine ladbare Datenstruktur vorhanden.
-412	<b>C16ERR_PGIF_USER_LIMIT</b>	Die maximale Anzahl der Benutzer ist erreicht.
-413	<b>C16ERR_USER_INVALID</b>	Benutzer ungültig
	<b>C16ERR_USER_SELF</b>	Es wurde versucht den eigenen Benutzer aus der Datenbank zu entfernen.
-415	<b>C16ERR_USER_SHM</b>	Es wurde versucht einen Benutzer, der über das Protokoll Shared Memory kommuniziert, aus der Datenbank zu entfernen.
-416	<b>C16ERR_AREA_STANDBY</b>	Es wurde versucht eine Datenbank zu öffnen, die im Standby-Modus ist.
-417	<b>C16ERR_AREA_ROLLBACK</b>	Die Datenbank befindet sich gerade im Rollback. Sie kann zu einem späteren Zeitpunkt geöffnet werden.
-418	<b>C16ERR_AREA_LOCKED_ADMIN</b>	siehe <u>Datenbank ist durch den Administrator gesperrt</u>
-419	<b>C16ERR_AREA_LOCKED_OPERATION</b>	siehe <u>Datenbank ist durch eine Serviceoperation gesperrt</u>
-420	<b>C16ERR_AREA_LOCKED_DOWN</b>	siehe <u>Datenbank ist durch das Herunterfahren gesperrt</u>
-421	<b>C16ERR_AREA_LOCKED_STANDBY</b>	

## Kontakt

-422	C16ERR_AREA_LOCKED_ROLLBACK	siehe <u>Datenbank ist gesperrt</u> <u>(Standby-Modus)</u>
-423	C16ERR_AREA_LOCKED_OPEN	siehe <u>Datenbank ist gesperrt</u> <u>(Sperre für Rollback)</u>
-424	C16ERR_AREA_LOCKED_NO_STANDBY_OPEN	siehe <u>Datenbank ist gesperrt</u> <u>(Sperre für Login)</u>
-501	C16ERR_INFO_BLOCKSIZE_INVALID	<u>Datenbank ist gesperrt</u> <u>(Standby-System nicht verfügbar)</u>
-601	C16ERR_NO_FILE	Die angegebene Strukturgröße (InfoSize) ist ungültig.
-602	C16ERR_NO_SBR	Die angegebene Datei existiert nicht.
-603	C16ERR_NO_FLD	Der angegebene Teildatensatz existiert nicht.
-604	C16ERR_NO_KEY	Das angegebene Feld existiert nicht.
-605	C16ERR_NO_KEY_FLD	Der angegebene Schlüssel existiert nicht.
-606	C16ERR_NO_LINK	Das angegebene Schlüsselfeld existiert nicht.
-607	C16ERR_NO_LINK_FLD	Die angegebene Verknüpfung existiert nicht.
-608	C16ERR_NO_FLD_DATA	Das angegebene Verknüpfungsfeld existiert nicht.
-609	C16ERR_LINK_INVALID	Datei enthält keine Felder.
-610	C16ERR_LOAD_DS	Die angegebene Verknüpfung ist ungültig.
-611	C16ERR_PROC	Die Datenstruktur konnte nicht geladen werden.
-620	C16ERR_NO_ARGUMENT	Fehler in der Prozedur.
-621	C16ERR_ARGUMENT_UNDEFINED	Es ist kein Argument angegeben.
-622	C16ERR_ARGUMENT_READ_ONLY	Das Argument ist nicht definiert / instanziert.
-701	C16ERR_TEXT_NAME_INVALID	Das Argument kann nicht beschrieben werden.
-702	C16ERR_TEXT_RIGHTS	Textname leer oder länger als 20 Stellen.
-703	C16ERR_TEXT_UNDEFINED	Benutzerberechtigung für eine Textoperation ist unzureichend.
		Kein gültige Textinformation im Textpuffer vorhanden.

## Kontakt

<b>-801</b>	<b>C16ERR_SEL_FILE_INVALID</b>	Die angegebene Datei existiert nicht
<b>-802</b>	<b>C16ERR_SEL_NAME_INVALID</b>	Selektionsname leer oder länger als 20 Zeichen
<b>-851</b>	<b>C16ERR_BIN_NAME_INVALID</b>	Objektname ist leer, länger als 60 Zeichen oder enthält ungültige Zeichen.
<b>-852</b>	<b>C16ERR_BIN_NO_PATH</b>	Pfad nicht vorhanden.
<b>-853</b>	<b>C16ERR_BIN_NO_DATA</b>	Keine Daten vorhanden.
<b>-854</b>	<b>C16ERR_BIN_DATA_ERR</b>	Falsches Codewort angegeben oder allgemeiner Fehler aufgetreten.
<b>-855</b>	<b>C16ERR_BIN_DECRYPTION</b>	Falsches Codewort angegeben.
<b>-856</b>	<b>C16ERR_BIN_OPERATION</b>	Operation kann nicht ausgeführt werden.
<b>-857</b>	<b>C16ERR_BIN_DIR_NOT_EMPTY</b>	Verzeichnis ist nicht leer.
	<b>C16ERR_DIAG_DEFECT</b>	Ergebnis nach einer Diagnose: Datenbank "DEFEKT".
<b>-901</b>		Ergebnis nach einer Diagnose: Datenbank "REPARIERT".
<b>-902</b>	<b>C16ERR_DIAG_REPAIRED</b>	Ergebnis nach einer Diagnose: Datenbank "KORRIGIERT".
<b>-903</b>	<b>C16ERR_DIAG_CORRECTED</b>	Diagnose läuft bereits.
<b>-904</b>	<b>C16ERR_DIAG_STARTED</b>	Es steht nicht genügend Hauptspeicher für eine Diagnose zur Verfügung.
	<b>C16ERR_DIAG_MEMORY</b>	Die Datenbank kann nur gelesen werden.
<b>-906</b>	<b>C16ERR_DIAG_READ_ONLY</b>	Aufruf der Diagnose-Befehle in der falschen Reihenfolge.
<b>-907</b>	<b>C16ERR_DIAG_SEQUENCE</b>	sonstiger Fehler bei der Diagnose.
<b>-908</b>	<b>C16ERR_DIAG_OTHER</b>	Die Datei c16.vra ist nicht vorhanden.
<b>-1001</b>	<b>C16ERR_VRA_MISSING</b>	Die Datei c16.tla ist nicht vorhanden.
<b>-1100</b>	<b>C16ERR_PCDT_UNDEF</b>	Die Version der Datei c16.tla ist inkompatibel.
<b>-1101</b>	<b>C16ERR_PCDT_VERSION</b>	Fehler beim Einlesen der Datei c16.tla.
<b>-1102</b>	<b>C16ERR_PCDT_READ</b>	Speicher-Allokation beim Einlesen der c16.tla
<b>-1102</b>	<b>C16ERR_PCDT_NOMEM</b>	

**Kontakt**

**fehlgeschlagen.**

## Kontakt

### vERROR

Rückgabewert der Funktionen

Siehe Fehlerliste

Alle Funktionen der CONZEPT 16-Programmierschnittstelle liefern ein Ergebnis vom Typ vERROR (signed int) zurück. Bei einem Rückgabewert = 0 wurde die Funktion erfolgreich ausgeführt. Im Falle eines negativen Wertes ist ein Fehler aufgetreten, der anhand des genauen Rückgabewertes Informationen zur Fehlerursache liefert. Eine Liste aller Rückgabewerte befindet sich in der Fehlerliste.

## Kontakt

### Die ODBC-Schnittstelle

Beschreibung der CONZEPT 16-ODBC-Schnittstelle Siehe

#### Blog

Mit Hilfe der CONZEPT 16-ODBC-Schnittstelle kann von einem beliebigen Programm aus eine Verbindung zum CONZEPT 16-Server aufgebaut werden. Die Verbindung erfolgt über das einheitliche Interface ODBC.

ODBC (Open Data Base Connectivity) ist eine standardisierte Schnittstelle zu Datenbanken, die SQL als Datenbanksprache verwendet. Welche SQL-Befehle unterstützt werden, hängt vom jeweiligen ODBC-Treiber ab.

Alle Anfragen, die abgeschickt werden, werden vom ODBC-Interface abgefangen und über zugehörige ODBC-Treiber an das Ziel-Datenbanksystem geschickt. Das Betriebssystem bzw. die Netzwerksoftware stellen eine weitere Zwischenstufe dar, welche die Geschwindigkeit beeinflussen kann.

Im Betriebssystem ist seit Windows 2000 auf 32-Bit-Systemen der ODBC-Datenquellen-Manager in der 32-Bit-Version installiert. Bei 64-Bit-Systemen ist sowohl die 32-Bit-Version, als auch die 64-Bit-Version installiert. Soll eine 32-Bit-Applikation auf den Treiber zugreifen, muss eine Datenquelle im 32-Bit-Manager eingetragen werden, für 64-Bit-Applikationen im 64-Bit-Manager.

Informationen zum generellen Aufbau der ODBC-Schnittstelle befinden sich im Abschnitt Aufbau der ODBC-Schnittstelle. Bevor eine CONZEPT 16-Datenbank als Datenquelle eingerichtet werden kann, muss die CONZEPT 16 ODBC-Schnittstelle installiert werden. Die Vorgehensweise bei der Installation ist im Abschnitt Installation der ODBC-Schnittstelle beschrieben. Danach kann gemäß der Beschreibung Datenquellen einrichten eine ODBC-Datenquelle eingerichtet werden.

## Kontakt

### ODBC

#### Open Database Connectivity

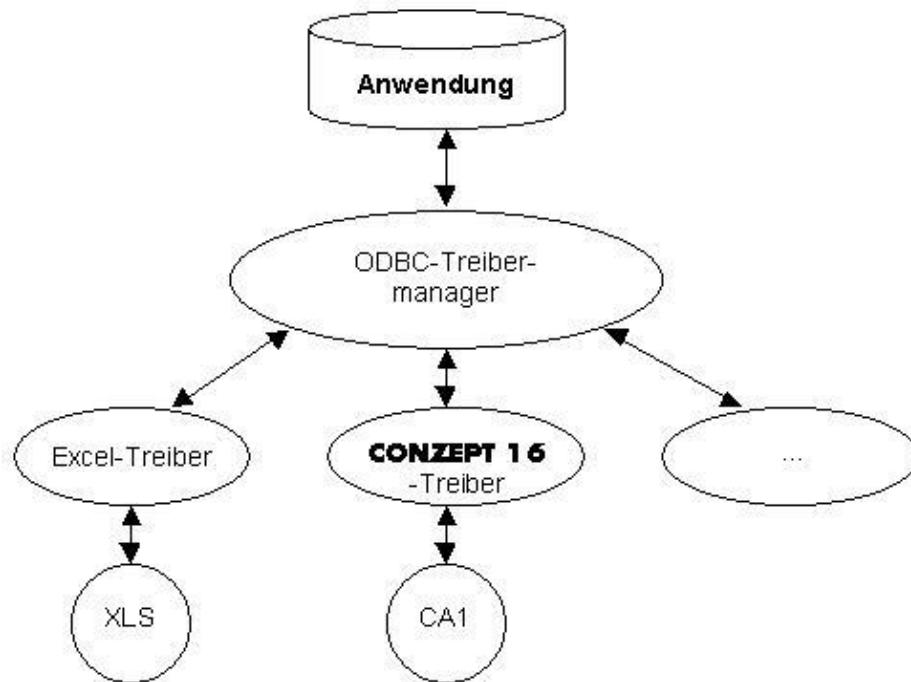
In CONZEPT 16 wird eine Schnittstelle bereit gestellt, mit der beliebige Applikationen auf den Datenbestand der Datenbank zugreifen können, die ODBC-Schnittstelle. Um die Schnittstelle nutzen zu können, muss der ODBC-Treiber auf dem Rechner installiert werden, von dem aus auf die CONZEPT 16-Datenbank zugegriffen werden soll.

Der Zugriff von CONZEPT 16 auf andere Datenbankmanagementsysteme über deren Datenquellen ist mit den Befehlen für ODBC-Verbindungen ebenfalls möglich.

## Kontakt

### Die ODBC-Schnittstelle - Aufbau Beschreibung der Kommunikation über ODBC

Für den Betrieb der Schnittstelle ist zum einen das Vorhandensein eines universellen Treibermanagers notwendig. Dieser stellt den jeweiligen Applikationen eine standardisierte Schnittstelle zur Verfügung. Für den Zugriff auf unterschiedliche Datenbanken wird jeweils ein entsprechender ODBC-Treiber benötigt, der für eine Reihe von Datenbanksystemen und anderen Programmen wie Tabellenkalkulationsprogramme verfügbar ist.



Die ODBC-Schnittstelle stellt eine Sammlung von Funktionsaufrufen bereit, die es erlauben, Verbindungen zu Datenbanksystemen herzustellen, SQL-Anweisungen durchzuführen und die Ergebnisse abzufragen. Die verwendete SQL-Syntax basiert auf der SQL-CAE-Spezifikation von X/Open und SQL Access Group (SAG) aus dem Jahre 1992.

Zusätzlich bietet sie ergänzende oder weiterführende Funktionen wie z. B. "Scrollable Cursors".

Es existieren zwei unterschiedliche Varianten der ODBC-Treiber: bei der Single-Tier-Konfiguration führt der Treiber die SQL-Anweisungen auf der Datenbankdatei direkt aus, bei der Multiple-Tier-Konfiguration leitet er lediglich die Anweisungen an einen SQL-Server weiter. Im Falle des CONZEPT 16-ODBC-Treibers werden die Anweisungen an den CONZEPT 16-Server weitergeleitet. Mit Hilfe von Data Sources (Datenquellen) können die Konfigurationen von unterschiedlichen Datenbanken in unbegrenzter Anzahl gespeichert werden und ermöglichen somit einen vereinfachten Verbindungsauflauf.

Innerhalb einer Netzwerkumgebung werden die ODBC-Anfragen des Clients zum Server übermittelt. Dort werden sie von einem ODBC-konformen Modul entgegengenommen und in Datenbankzugriffe umgesetzt. Der Datenbank-Server kann

## **Kontakt**

**auf anderen Rechnersystemen installiert sein.**

**ODBC-Objekte**

**Liste der ODBC-Objekte**

- OdbcApi
- OdbcClm
- OdbcCon
- OdbcStm
- OdbcTbl

## Kontakt

OdbcApi



ODBC-Schnittstellen-Objekt

Siehe [Befehle](#),

[Eigenschaften](#)

Das Objekt wird durch die Anweisung [OdbcOpen\(\)](#) erzeugt. Über die Eigenschaften des Objekts können Informationen über die CONZEPT 16-ODBC-Schnittstelle ermittelt werden.

Das Objekt bleibt bestehen, bis es mit [OdbcClose\(\)](#) geschlossen wird.

## Kontakt

OdbcClm



ODBC-Spalten-Objekt

Siehe [Befehle](#),

[Eigenschaften](#)

Das Objekt wird durch die Anweisung [OdbcCatalogClm\(\)](#) erzeugt. Die Eigenschaften des Objekts enthalten die Attribute einer Spalte. Die Attribute der nächsten Spalte können mit der Anweisung [OdbcFetch\(\)](#) gelesen werden.

Das Objekt bleibt bestehen, bis es mit [OdbcClose\(\)](#) entfernt, oder die Verbindung zur Datenquelle getrennt wird.

## Kontakt

OdbcCon



ODBC-Verbindungs-Objekt

Siehe [Befehle](#),

[Eigenschaften](#)

Das Objekt wird durch die Anweisung [OdbcConnect\(\)](#) und [OdbcConnectDriver\(\)](#) erzeugt.

Über die Eigenschaften des Objekts können Informationen über den ODBC-Treiber und die verbundene Datenbank ermittelt werden.

Das Objekt bleibt bestehen, bis es mit [OdbcClose\(\)](#) entfernt, oder die ODBC-Schnittstelle geschlossen wird.

## Kontakt

OdbcStm



ODBC-Statement-Objekt

Siehe [Befehle](#),

[Eigenschaften](#)

Das Objekt wird durch die Anweisung [OdbcPrepare\(\)](#) und [OdbcExecuteDirect\(\)](#) erzeugt. Über die Eigenschaften des Objekts können Informationen zu dem Statement oder seiner Ergebnismenge ermittelt werden. Das Objekt wird der Anweisung [OdbcFetch\(\)](#) übergeben, um die Ergebnismenge auszulesen.

Das Objekt bleibt bestehen, bis es mit [OdbcClose\(\)](#) entfernt, oder die Verbindung zur Datenquelle getrennt wird.

### OdbcTbl



### ODBC-Tabellen-Objekt

Siehe [Befehle](#),

#### [Eigenschaften](#)

Das Objekt wird durch die Anweisung [OdbcCatalogTbl\(\)](#) erzeugt. Die Eigenschaften des Objekts enthalten die Attribute einer Tabelle. Die Attribute der nächsten Tabelle können mit der Anweisung [OdbcFetch\(\)](#) gelesen werden.

Das Objekt bleibt bestehen, bis es mit [OdbcClose\(\)](#) entfernt, oder die Verbindung zur Datenquelle getrennt wird.

## Kontakt

### Installation der ODBC-Schnittstelle

#### Beschreibung der Installation der ODBC-Schnittstelle von CONZEPT 16

Die ODBC-Schnittstelle wird über die CONZEPT 16 Installationsroutine installiert. Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \Odbc8. Über die Installationsroutine kann der ODBC-Treiber installiert werden. ODBC-Clients können lokal über diesen auf eine CONZEPT 16-Datenbank zugreifen. Diese Installation ist für den normalen Betrieb ausreichend.

Eine Beschreibung des ODBC-Treibers befindet sich im Abschnitt ODBC-Treiber.

### Programmdateien

#### *Dateien der ODBC-Schnittstelle*

\w32\Driver  
c16\_odbc\_driver.dll  
c16\_odbc\_driver\_messages\_en-US.xml  
ODBCMessages\_en-US.xml  
simbaicudt53\_32.dll  
simbaicuin53\_32.dll  
simbaicuuc53\_32.dll  
SQLEngineMessages\_en-US.xml  
\w64\Driver  
c16\_odbc\_driver.dll  
c16\_odbc\_driver\_messages\_en-US.xml  
ODBCMessages\_en-US.xml  
simbaicudt53\_64.dll  
simbaicuin53\_64.dll  
simbaicuuc53\_64.dll  
SQLEngineMessages\_en-US.xml



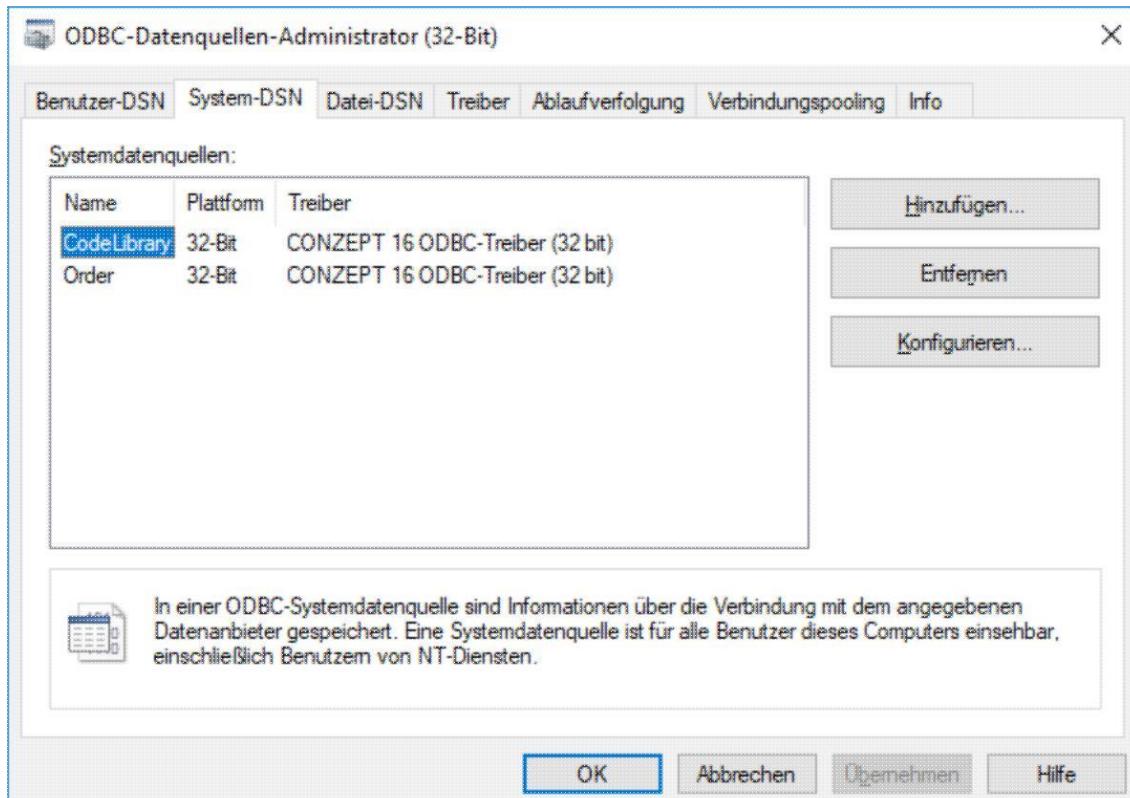
Nach der Installation des Programmstandes müssen die CONZEPT 16 Datenquellen, welche vom ODBC-Treiber zur Verfügung gestellt werden sollen, beim Windows ODBC-Treiber Manager bekanntgegeben werden. Siehe dazu ODBC-Datenquellen einrichten.

## Kontakt

### Die ODBC-Schnittstelle - Datenquellen einrichten

#### Einrichten von Datenquellen

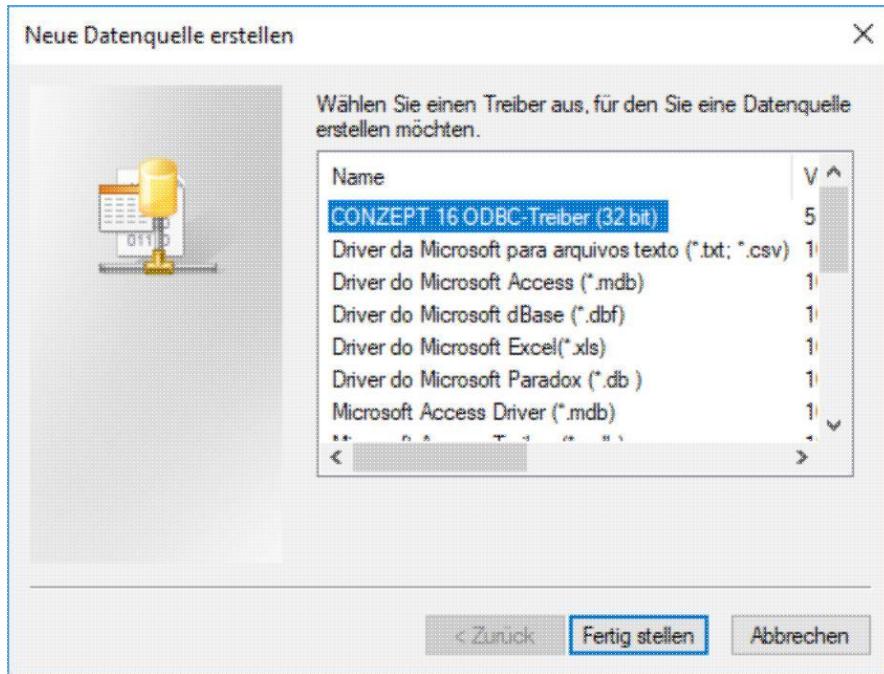
Das Einrichten von Datenquellen erfolgt im ODBC-Datenquellen-Administrator. Bevor eine Datenquelle über den ODBC-Treibermanager angesprochen werden kann, muss der entsprechende ODBC-Treiber installiert und die Datenquelle dem Manager bekanntgegeben werden. Der Data-Source-Name-Manager (ODBC Datenquellen Administrator) wird über die Systemsteuerung im Bereich Verwaltung gestartet. Datenquellen können in den Registerkarten Benutzer-DSN, System-DSN und Datei-DSN eingetragen werden. Die Datenquellen sind je nach Eintragung nur von einem Benutzer, von allen Benutzern des Systems oder von Netzwerkbenutzern zugänglich.



Über die Schaltfläche [Hinzufügen...] können neue Datenquellen eingerichtet werden.

Nach dem Drücken der Schaltfläche erscheint folgender Auswahldialog:

## Kontakt



In der Liste wird der gewünschte ODBC-Treiber ausgewählt und die Schaltfläche [Fertig stellen] angeklickt.

**i** Unter 64-Bit-Systemen zeigt die Liste kaum einen ODBC-Treiber an. Hier werden nur die 64-Bit-Treiber angezeigt. Um den ODBC-Treibermanager für die 32-Bit-Treiber zu starten, muss aus dem Windows-Verzeichnis, aus dem Unterverzeichnis SysWoW64 das Programm OdbcAd32.exe gestartet werden. Von 64-Bit-Programmen können nur 64-Bit-Datenquellen geladen werden, von 32-Bit-Programmen nur 32-Bit-Datenquellen.

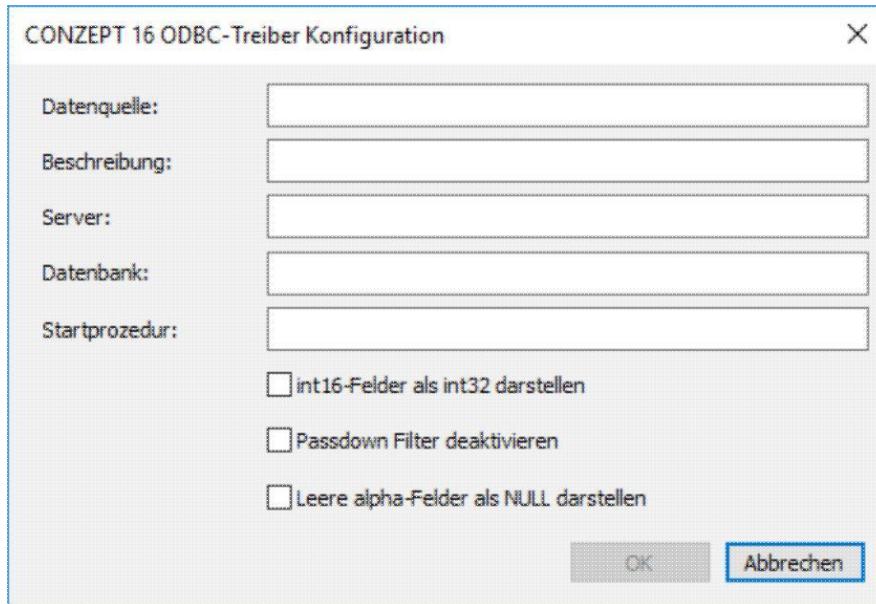
Die weitere Vorgehensweise hängt vom ausgewählten ODBC-Treiber ab:

- CONZEPT 16-ODBC-Treiber
- Andere ODBC-Treiber

### CONZEPT 16-ODBC-Treiber

Um eine CONZEPT 16-Datenbank eintragen zu können, muss die CA1-Datei vorhanden und beim CONZEPT 16-Server eingetragen sein. Nach dem Drücken der Schaltfläche [Fertig stellen] erscheint folgender Dialog:

## Kontakt



**Der CONZEPT 16-Treiber benötigt für die Datenquelle einen Namen, der im Feld "Datenquelle" eingetragen wird. Über den Namen der Datenquelle greifen später die Programme auf die Datenbank zu. Danach wird der Name oder die IP-Adresse des CONZEPT 16-Datenbank-Servers im Feld "Server" eingetragen.**

**Im Feld "Datenbank" wird der Name der Datenbank oder der Pfad und der Dateiname (ohne Dateierweiterung .CA1) angegeben. Die Eintragung entspricht der Eintragung beim CONZEPT 16-Server.**

**Die Angabe einer Beschreibung und einer Startprozedur ist optional und kann entfallen. Nähere Informationen zur Startprozedur befinden sich im Abschnitt Funktionen.**

**Einige Programme können Werte aus ganzzahlig kurzen Feldern (16-Bit Integer ohne Vorzeichen) nicht korrekt verarbeiten. Damit in diesen Fällen trotzdem ein vernünftiger Zugriff auf die Daten möglich ist, kann durch einen Haken bei "int16-Felder als int32 darstellen" erreicht werden, dass sich ganzzahlig kurze Felder nach aussen als ganzzahlig lang (32-Bit mit Vorzeichen) darstellen. In diesen Feldern können nach wie vor nur Werte im Bereich von 0 bis 65535 gespeichert werden.**

**Bei Verwendung der Passdown Filter werden die Filterkriterien der SELECT-, UPDATE- und DELETE-Statements komplett an den Server geschickt und von diesem verarbeitet. Dieser gibt nur die passenden Datensätze an den ODBC-Treiber zurück. Über "Passdown Filter deaktivieren" kann die Passdown Technologie abgeschaltet werden. Dadurch werden bei einem Statement die Daten durch den ODBC-Treiber komplett vom Server gelesen und vom Treiber verarbeitet und ausgewertet.**

**Mit der Einstellung "Leere alpha-Felder als NULL darstellen" wird definiert, dass leere alpha-Felder als NULL-Wert übertragen werden.**

**Im folgenden einige Beispiele für die Definition von Datenquellen, die sich mit einem CONZEPT 16-Server auf unterschiedlichen Betriebssystemen verbinden:**

## Kontakt

### Windows-Server Linux-Server

Protokoll	TCP	TCP
Server	127.0.0.1	DBSERV
Datenbank Auftrag		/c16/db/auftrag

Im Falle des Windows-Servers wurde der symbolische Namen der Datenbank angegeben, bei Linux der komplette Datenbankpfad. Unter beiden Plattformen sind beide Angaben möglich. Mit der ODBC-Schnittstelle kann ebenfalls die HotStandby-Option des Servers genutzt werden. Im Eingabefeld "Server" werden dann beide IP-Adressen der Server durch ein Plus-Zeichen getrennt angegeben.

Bei der Verwendung einer Datei-DSN müssen in der erzeugten Datei folgende Einträge vorhanden sein:

Eintrag	Bemerkung
[ODBC]	
DRIVER=CONZEPT 16 ODBC-Treiber (32 bit)	Name des CONZEPT 16-ODBC-Treibers
SERVER=<servername>	Servername oder IP-Adresse
DB=<db-name>	Name der Datenbank
STARTPROC=<prozedurname>	Startprozedur (optional)
UID=<user-name>	Benutzername (optional)

Ein Datei-DSN kann beispielsweise im Netzwerk verwendet werden, um von Programmen auf unterschiedlichen Rechnern auf die selbe Datenquelle zuzugreifen, ohne dass die Datenquelle auf allen Rechnern eingerichtet werden muss. Lediglich der Treiber muss auf allen Rechnern vorhanden sein.

### Andere ODBC-Treiber

Die Definition einer Datenquelle ist für jeden ODBC-Treiber anders. Die notwendigen Eintragungen in der Konfiguration müssen der Dokumentation des entsprechenden ODBC-Treibers entnommen werden.

## Kontakt

### Zugriff auf Datenquellen

#### Verwendung der ODBC-Schnittstelle aus CONZEPT 16

Um vom CONZEPT 16-Client aus externe Datenquellen ansprechen zu können, muss eine entsprechende Datenquelle beim ODBC-Datenquellen-Manager eingerichtet sein (vgl. [Datenquellen einrichten](#)). Damit die entsprechenden ODBC-Befehle verwendet werden können, muss eine Enterprise- oder Developer-Edition installiert sein.

### Befehlssatz

Eine Liste der Befehle, über die eine Datenquelle geöffnet und auf deren Daten zugegriffen werden können, befindet sich im Abschnitt [Befehle für ODBC-Verbindungen](#).

## Kontakt

### Die ODBC-Schnittstelle - Der ODBC-Treiber

#### Beschreibung des CONZEPT 16-ODBC-Treibers

 Eine Auflistung der Betriebssysteme, auf denen der ODBC-Treiber betrieben werden kann, finden Sie in den Systemvoraussetzungen.

Mit der Installation des ODBC-Treibers für CONZEPT 16 ist es möglich, dass fremde Programme auf die Datenbestände der CONZEPT 16-Datenbank zugreifen können. Zum Einsatz des ODBC-Treibers wird ein CONZEPT 16-Server ab Release 1.5.06 benötigt.

Um auf eine CONZEPT 16-Datenbank zuzugreifen, kann zum Beispiel das Programm WinSQL von Synametrics Technologies verwendet werden. Es kann aber auch jedes andere Programm verwendet werden, dass in der Lage ist, über die ODBC-Schnittstelle zu kommunizieren.

### CONZEPT 16-ODBC-Treiber

Bei der Verwendung des CONZEPT 16-ODBC-Treibers werden die ODBC-Befehle vom Treiber übersetzt und die Filterkriterien des Statements an den CONZEPT 16-Server geschickt und von diesem verarbeitet. Der Treiber erhält nur die passenden Datensätze vom Server zurück.

Wird die Verwendung der Passdown Filter deaktiviert, werden alle Datensätze vom Server an den Treiber zurückgeschickt und vom Treiber zur Übermittlung an die Anwendung weiter aufbereitet.

Die Anwendung kann schließlich die Daten abholen und weiter verarbeiten. In dieser Konstellation muss die Datenquelle auf der gleichen Maschine angelegt sein, auf der auch die Anwendung läuft. Der CONZEPT 16-Server und die Datenbank können auf einem anderen System laufen.

Die weitere Vorgehensweise wird in folgenden Abschnitten erläutert:

- Datenzugriff
- Funktionen
- Verbindung mit der Datenquelle
- Sprachumfang
- Fehlermeldungen

## Kontakt

### Die ODBC-Schnittstelle - Datenzugriff

#### Voraussetzungen für den Datenzugriff

Um von einer beliebigen Anwendung auf eine CONZEPT 16-Datenbank zugreifen zu können müssen folgende Voraussetzungen erfüllt sein:

##### 1. Benutzer mit externen Zugriffsrechten

Die Anwendung muss sich mit einem Datenbankbenutzer an der Datenquelle anmelden, der für den "Externen Zugriff" freigegeben ist.

##### 2. Dateirechte des Benutzers

Die Anwendung kann nur auf die Dateien zugreifen, für die der Datenbankbenutzer ausreichende Berechtigungen besitzt.

##### 3. Berechtigung Textsystem

Die Anwendung kann ebenfalls lesend auf interne Texte der Datenbank zugreifen, wenn der Datenbankbenutzer über entsprechende Berechtigungen für das Textsystem verfügt.

Versucht sich eine Anwendung mit einem Benutzer anzumelden, der keinen externen Zugriff erlaubt, wird die Anmeldung abgelehnt. Ebenso wird bei einem Zugriff auf Dateien oder Texte ohne ausreichende Berechtigungen ein entsprechender Fehler (siehe Fehlermeldungen) zurückgegeben.

### Zugriff auf interne Texte

Über die ODBC-Schnittstelle kann neben den Tabellen ebenfalls auf interne Texte der Datenbank zugegriffen werden. Dazu steht eine Tabelle mit dem Namen Text zur Verfügung. Der Zugriff auf die internen Texte beschränkt sich auf das Lesen der Texte.

In dieser Tabelle sind folgende Spalten enthalten:

Feldname	SQL-Typ	Verwendung
Text_Keyname	SQL_VARCHAR(20)	Diese Spalte enthält den Namen des Textes in Großbuchstaben. Über diese Spalte ist ein Index definiert. Zugriffe auf Texte sollten deshalb über diese Spalten gehen.
Text_Name	SQL_VARCHAR(20)	Name des Textes.
Text_Group	SQL_VARCHAR(20)	Name der Textgruppe.
Text_UserPrivate	SQL_VARCHAR(20)	Name des Benutzers, falls der Text privat gespeichert wurde.
Text_UserModify	SQL_VARCHAR(20)	Name des Benutzers, der den Text zuletzt gespeichert hat.
Text_CreateDate	SQL_DATE	Erstellungsdatum des Textes.
Text_CreateTime	SQL_TIME	Erstellungszeit des Textes.
Text_ModifyDate	SQL_DATE	Datum der letzten Änderung.
Text_ModifyTime	SQL_TIME	Zeit der letzten Änderung.

## Kontakt

<b>Text_ID</b>	<b>SQL_INTEGER</b>	ID-Nummer des Textes. Die ID ist immer negativ.
<b>Text_Size</b>	<b>SQL_INTEGER</b>	Größe des Textes in Bytes.
<b>Text_Lines</b>	<b>SQL_INTEGER</b>	Anzahl der Zeilen.
<b>Text_Private</b>	<b>SQL_BIT</b>	Text wurde als privat gespeichert.
<b>Text_Encrypted</b>	<b>SQL_BIT</b>	Text wurde chiffriert gespeichert.
<b>Text_Data</b>	<b>SQL_LONGVARCHAR</b>	Textinhalt. Einzelne Zeilen sind durch <b>&lt;CR&gt;&lt;LF&gt;</b> voneinander getrennt.

Größere Texte werden von verschiedenen ODBC-Clients nicht vollständig gelesen, da die Clients meist eine Größenbeschränkung für Daten vom Typ SQL\_LONGVARCHAR haben.

Bei Texten, für die der Benutzer nicht über ausreichende Leseberechtigung verfügt, wird ein leerer Textinhalt zurückgeliefert.

## Kontakt

### Die ODBC-Schnittstelle - Funktionen

#### Aufruf von Funktionen

Beim Einrichten einer Datenquelle kann eine Startprozedur angegeben werden.

Diese A+-Prozedur (Übersetzt mit der Compileroption @A+) wird beim Einloggen eines Benutzers über die ODBC-Schnittstelle ausgeführt. In dieser Prozedur können mit dem Befehl FldAttributes() Feldattribute gesetzt und mit ProcAdvertise() Prozeduren bekannt gemacht werden.

Die Feldattribute werden benötigt, wenn die Datensätze nach bestimmten Regeln sortiert übermittelt werden sollen. Sollen die Datensätze nach einem alphanumerischen Feld sortiert werden, erfolgt die Sortierung nach der ASCII-Tabelle. In der Sortierung sind also alle Großbuchstaben vor den Kleinbuchstaben sortiert. Soll aber zum Beispiel die Groß- und Klein-Schreibung nicht berücksichtigt werden, können entsprechende Feldattribute mit der Anweisung FldAttributes() gesetzt werden (siehe dort).

#### Zugriff auf Funktionen

Über die ODBC-Schnittstelle können ebenfalls Funktionen aus CONZEPT 16 aufgerufen werden. Diese "Stored Procedures" müssen zuvor mit dem Befehl ProcAdvertise() bekannt gemacht werden. Die Bekanntgabe der Funktionen kann entweder in der Startprozedur der ODBC-Datenquelle oder in einer Stored Procedure erfolgen.

Eine Beschränkung der Benutzer auf bestimmte Funktionen kann somit durch die Bekanntgabe einer Anmeldefunktion realisiert werden. Dieser Funktion werden Benutzer und Passwort übergeben. Die Überprüfung der Angaben erfolgt innerhalb von CONZEPT 16. In Abhängigkeit des Benutzers können dann unterschiedliche Funktionen bekanntgegeben werden. Funktionen, die nicht bekannt gegeben wurden, können über die Schnittstelle nicht aufgerufen werden.

Innerhalb der Funktionen können keine Dialoge oder Fenster geöffnet werden.

Ergebniswerte können entweder durch den Rückgabewert der Funktion oder durch die Übergabe eines var-Arguments (Call-by-Reference) erfolgen. Als Übergabe- und Rückgabewerte können nur Datentypen verwendet werden, die sowohl in CONZEPT 16, als auch in SQL bekannt sind. Die Übergabe einer Variablen vom Typ font oder von einem Array ist zum Beispiel nicht möglich. Weitere Informationen befinden sich in der Beschreibung des Befehls ProcAdvertise().

## Kontakt

### ODBC-Befehle

Befehle, die in der Startprozedur einer ODBC-Datenquelle durchgeführt werden können

Verwandte

Befehle, Liste

sortiert nach

Siehe Gruppen,

Alphabetische

Liste aller

Befehle

Befehle

- FldAttributes

- ProcAdvertise

## Kontakt

Die ODBC-Schnittstelle - Verbindung mit der Datenquelle Verbindung mit einer CONZEPT 16-Datenbank als Datenquelle

Bevor ein Benutzer eine Datenquelle öffnen kann, wird von dem ODBC-Treiber ein Anmeldedialog angezeigt:



Im Feld "Benutzer" muss ein in der Datenbank eingetragener Benutzer angegeben werden. Ist dieser Benutzer mit einem Passwort geschützt, erfolgt die Passworteingabe im Feld "Passwort". Es kommt nur dann eine Verbindung zustande, wenn in mindestens einer Benutzergruppe, der der Benutzer angehört, die Option "Externer Zugriff" gesetzt ist.

## Kontakt

### Benutzerverwaltung

#### Benutzergruppen

Name
DEV
EXE
PLUGIN
START
WEB
_Administrators
_Everyone

#### Programmrechte

- |   |   |
|---|---|
| <input checked="" type="checkbox"/> Konfiguration   | <input type="checkbox"/> Basisparameter                   |
| <input checked="" type="checkbox"/> Textsystem      | <input checked="" type="checkbox"/> Prozeduren            |
| <input checked="" type="checkbox"/> Testfunktionen  | <input checked="" type="checkbox"/> Menüs                 |
| <input checked="" type="checkbox"/> Notizen         | <input checked="" type="checkbox"/> Druckertreiber        |
| <br>  | <br>  |
| <input checked="" type="checkbox"/> Datensicherung  | <input checked="" type="checkbox"/> DB-Info im Textsystem |
| <input checked="" type="checkbox"/> Datenbankpflege | <input checked="" type="checkbox"/> Externer Zugriff      |

Berechtigung Menüs

0 ▲ ▼

Berechtigung Textsystem

0 ▲ ▼

Berechtigung Druckertreiber

0 ▲ ▼

 Übernehmen

 Abbrechen

 Benutzergruppen

WEB

Der Benutzer bekommt darüber hinaus nur die Dateien angezeigt, für die der Benutzer eine Berechtigung größer als 0 besitzt.

 Datenbanken der Version 4.0 können mit der ODBC-Schnittstelle ebenfalls geöffnet werden. In diesen Datenbanken kann allerdings der externe Zugriff nicht unterbunden werden. Sobald ein Benutzer über ein gültiges Kennwort verfügt, kann die Datenbank über die ODBC-Schnittstelle geöffnet werden.

Darüber hinaus werden folgende Dateiberechtigungen ausgewertet:

## Kontakt

- **Zugriff** - Der Zugriff auf die Daten der Datei ist gewährleistet, sobald der Benutzer eine Berechtigung von größer als 0 und für den externen Zugriff besitzt. Das Zugriffsrecht wird an dieser Stelle nicht ausgewertet.
- **Speichern** - Ist dieses Recht gesetzt, können mit dem INSERT-Statement neue Datensätze eingefügt werden.
- **Ändern** - Vorhandene Datensätze können mit dem UPDATE-Statement verändert werden.
- **Löschen** - Vorhandene Datensätze können mit dem DELETE-Statement gelöscht werden.

## Kontakt

### Die ODBC-Schnittstelle - Sprachumfang

Beschreibung des Sprachumfanges des CONZEPT 16-ODBC-Treibers Es ist der Sprachumfang nach ODBC 3.52 Level 1 und 2 implementiert.

### Datentypen

Folgende Datentypen werden unterstützt:

CONZEPT	SQL-Datentyp
<u>16-Datentyp</u>	
<u>alpha</u>	CHARACTER VARYING(n) oder VARCHAR(n) n maximal 8192
<u>bigint</u>	BIGINT
<u>int</u>	INTEGER
<u>word</u>	SMALLINT
<u>decimal</u>	DECIMAL maximale Genauigkeit von 58 signifikanten Stellen
<u>float</u>	DOUBLE
<u>logic</u>	BIT
<u>time</u>	TIME
<u>date</u>	DATE

### Unterstützter Befehlsumfang

Aus Gründen der Kompatibilität der Datenstruktur zu bestehenden Datenbank-Clients sind nur die Befehle zur Datenmanipulation (DML) implementiert.

## Kontakt

### Die ODBC-Schnittstelle - Fehlermeldungen

Beschreibung der von der CONZEPT 16-ODBC-Schnittstelle zurückgegebenen Fehlermeldungen

Folgende Fehlermeldungen werden vom ODBC-Treiber zurückgegeben:

Communication failure	Es trat ein Fehler bei der Kommunikation zwischen der ODBC-Schnittstelle und dem CONZEPT 16-Server auf.
Connect to server failed	Es konnte keine Verbindung zum Server aufgenommen werden.
Database is locked	Datenbank ist gesperrt.
Database is locked by administrator	Siehe <a href="#">Datenbank ist durch den Administrator gesperrt</a>
Database is locked by shutdown	Siehe <a href="#">Datenbank ist durch das Herunterfahren gesperrt</a>
Database is locked by service operation	Siehe <a href="#">Datenbank ist durch eine Serviceoperation gesperrt</a>
Database is locked (standby mode)	Siehe <a href="#">Datenbank ist gesperrt (Standby-Modus)</a>
Database is locked (rollback denied)	Siehe <a href="#">Datenbank ist gesperrt (Sperre für Rollback)</a>
Database is locked (login denied)	Siehe <a href="#">Datenbank ist gesperrt (Sperre für Login)</a>
Database is locked (standby server not available)	Siehe <a href="#">Datenbank ist gesperrt (Standby-System nicht verfügbar)</a>
Database is protected	Die Datenbank wurde mit einem zusätzlichen Kennwort geschützt. Es wurde innerhalb einer Session versucht die Datenbank ein zweites Mal zu öffnen.
Database limit reached	Die in der Konfiguration angegebene Datenbank wurde nicht gefunden.
Database not found	Die Datenbank konnte nicht geöffnet werden. Die Datenbank ist vom Server in exklusiver Benutzung (z. B. Recover).
Database open failed	Allgemeiner Fehler
Database unavailable	Die maximale Anzahl der Benutzer der ODBC-Schnittstelle stimmt nicht mit der Benutzeranzahl des CONZEPT 16-Servers überein.
Generic error	Es wurde versucht eine Operation durchzuführen, für die der Benutzer keine ausreichende Berechtigung besitzt.
Insufficient number of user licenses	Die Datenbank hat nicht den korrekten Versionsstand.
Insufficient user rights	Bei der Anmeldung wurde ein falscher Benutzer oder ein falsches Kennwort angegeben.
Invalid type of database	Der Zeitraum für die ODBC-Evaluierungslizenz ist abgelaufen.
Invalid user account or password	
License expired	
Record lock violation	

## Kontakt

<b>Start of server process failed</b>	<b>Es wurde versucht einen gesperrten Datensatz zu ändern oder zu löschen.</b>
<b>Unique key violation</b>	<b>Es konnte kein Server-Prozess gestartet werden (z. B. weil der Server gerade heruntergefahren wird, ggf. ein paar Sekunden warten und den Versuch wiederholen).</b>
<b>User limit is reached</b>	<b>Es wurde versucht einen Datensatz mit einem bereits in der Datenbank vorhandenen einmaligen Schlüsselwert zu speichern.</b> <b>Der Benutzer wurde bei der Anmeldung abgewiesen, da bereits die maximale Anzahl von Benutzern in der Datenbank eingeloggt sind.</b>

Siehe auch Hinweise im FAQ-Bereich!

Eine Reihe von weiteren Fehlermeldungen werden vom ODBC-Treibermanager und dem Programm, welches auf die Datenbank zugreift, erzeugt. Die Bedeutung dieser Fehlermeldungen sind der jeweiligen Beschreibung zu entnehmen.

## Kontakt

### Web-Schnittstelle - Web-Anbindung einer Datenbank

#### Beschreibung der Web-Schnittstelle

 Eine Auflistung der Betriebssysteme, auf denen die Web-Schnittstelle betrieben werden kann, finden Sie in den [Systemvoraussetzungen](#).

Die Web-Schnittstelle realisiert den Zugriff auf eine CONZEPT 16-Datenbank über einen Web-Browser. Für die Benutzung werden auf der Seite des Browsers keine Plug-Ins oder sonstigen Erweiterungen benötigt. Somit kann jeder Browser auf den Client zugreifen. Dieser stellt eine Erweiterung des Webservers dar und läuft unabhängig von anderen Websprachen. Aus dem Client heraus können bestehende Seiten verändert oder vollständige Seiten aus der Datenbank erzeugt werden. Dem Benutzer des Browsers bleibt dabei verborgen, woher die Seiten stammen.

In diesem Abschnitt werden die Möglichkeiten der Web-Schnittstelle beschrieben.

Folgende Themen werden behandelt:

- [Verarbeitungsweise der Web-Schnittstelle](#)
- [Installation der Web-Schnittstelle](#)
- [Einstellungen des Internet Information Services](#)
- [Einstellungen der c16\\_web.dll](#)
- [Einstellungen des Browsers](#)
- [Aufruf von Prozeduren von der Web-Schnittstelle](#)
- [Anweisungen im HTML-Dokument](#)
- [Befehle der Web-Schnittstelle](#)
- [Erstellung von HTML-Seiten](#)
- [Fehlermeldungen der Web-Schnittstelle](#)

## **Kontakt**

### **Web-Schnittstelle - Verarbeitungsweise Kommunikation im Internet und der Datenbank**

**Zur Kommunikation im Internet wird das Protokoll HTTP (Hypertext Transfer Protocol) verwendet. HTTP überträgt formatierte Texte und darin eingebettete Verbindungen zu weiteren Dokumenten. Die Referenz auf ein Dokument wird in Form einer URL (uniform resource locator) angegeben. Ein solcher Link spezifiziert das Protokoll, den Namen des Computers, den Pfad und den Namen der Datei (Seite), die dargestellt werden soll.**

#### **Beispiel:**

**http://www.vectorsoft.de/default.htm**

**Der formatierte Text wird von einem Browser auf dem Clientrechner interpretiert und dargestellt. Eine Speicherung des Textes muss dazu nicht erfolgen. Üblicherweise wird aus Gründen der Effizienz dennoch die Seite auf dem Clientrechner zwischengespeichert.**

**Weitere Informationen zum Datenaustausch befinden sich in den Abschnitten  
Kommunikation über HTTP und Kommunikation mit der Server-Erweiterung.**

## Kontakt

### Web-Schnittstelle - Kommunikation über HTTP

#### Kommunikation im Internet über HTTP

Das HTTP ist ein sogenanntes "Klartext"-Protokoll. Die Kommunikation erfolgt dabei über Zeichenfolgen, die auch vom Benutzer gelesen werden können. Das Protokoll verwendet ein System von Anfragen und Antworten. Nach dem Eintragen einer Web-Adresse in einem Browser wird der Name des Web-Servers in eine IP-Nummer umgesetzt (dies geschieht normalerweise durch einen DNS-Server) und eine Verbindung zum HTTP-Server aufgebaut. Bei dem Link <http://www.vectorsoft.de/default.htm> wird der Name www.vectorsoft.de in die IP-Adresse 62.132.119.100 umgesetzt. Der Verbindungsaufbau und die Kontrolle der Verbindung wird durch die unter HTTP liegende Protokollsicht (dem Internet-Protokoll TCP/IP) durchgeführt. Steht diese Verbindung, wird ein Request (Anfrage) an den Server gesendet, woraufhin dieser eine Response (Antwort) zurücksendet:

**Request    GET /default.htm HTTP/1.0**

**Response Inhalt der Datei DEFAULT.HTM**

Der Request beinhaltet mindestens die Anfragemethode (GET), die angeforderte Ressource (/default.htm) und die vom Browser unterstützte Protokollversion (HTTP/1.0). Nach der Übertragung der Ressource wird die Verbindung zwischen Browserrechner und Web-Server wieder abgebaut. Der hier dargestellte Fall ist stark vereinfacht, da oft andere Rechner zum Verbindungsaufbau beitragen, wie zum Beispiel Gateways oder Proxy-Server, bei denen der Request und Response zwischengespeichert und/oder eine Umsetzung von IP-Adressen stattfinden kann. Neben der Methode (beispielsweise GET) können weitere Informationen im Request übermittelt werden. Der Request-Header enthält in der Regel Informationen über den Browserrechner. Die Informationen sind in dem Format <Bezeichner>: <Wert> abgelegt. Die Bezeichner im Header sind nur zum Teil standardisiert und sind nicht zwingend für die Kommunikation notwendig. Der oben stehende Request könnte folgendermaßen aussehen:

```
GET /default.htm HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, */
UA-Pixels: 1024x768
UA-Color: color16
...
```

Die Informationen können vom Web-Server verwendet werden, um gegebenenfalls Rücksicht auf die Eigenschaften des Browsers und dessen Umgebung nehmen zu können. Neben der Methode GET sind noch weitere Request-Methoden definiert. Da vom Client nur die Methoden GET, POST und HEAD verarbeitet werden, soll an dieser Stelle noch auf die Methode POST eingegangen werden. Bei der Kommunikation zwischen Browser und Web-Server müssen häufig Benutzereingaben vom Browser an den Server übermittelt werden. Diese Daten werden z. B. in einer HTML-Seite in ein Formular eingetragen. In folgender Seite soll ein Name und Vorname eingetragen werden:

```
<html><head></head><body>
<form>
Name: <input name="Name"><br>
Vorname: <input name="Vorname"><br>
<input type="submit"><input type="reset">
</form>
</body></html>
```

## Kontakt

Die Formulardaten können sowohl mit der Methode GET als auch mit POST übertragen werden, die Methoden unterscheiden sich dabei in der Art der Datenübermittlung. Wird die Methode GET verwendet, werden die Daten in der URL mit übergeben. Wurde also beim Namen Doe und bei Vorname John eingegeben entsteht folgender Request:

```
GET /scripts/c16_web.dll?Name=Doe&Vorname=John HTTP/1.0<Header>
```

Die eingegebenen Informationen werden nach dem Fragezeichen, im sogenannten Query-String, hinter der Ressource angegeben. Bei Verwendung der Methode POST werden die Daten nach dem Header übertragen. Der Request würde also wie folgt aussehen:

```
POST /scripts/c16_web.dll HTTP/1.0<Header><Leerzeile>Name=DoeVorname=John
```

Die verwendete Methode kann im <form>-Tag angegeben werden: <form method=post>. Die Methode POST sollte verwendet werden, wenn das Formular größere Datenmengen beinhalten kann. Die Länge des Query-Strings ist auf 4 KB beschränkt, sollte aber nicht mehr als 1 KB beinhalten. Die Antwort auf ein Request des Browsers beinhaltet neben dem angeforderten Dokument auch einen Header. Dieser Header wird vom Browser ausgewertet und enthält Informationen über den Response-Body. Eine der wichtigsten Informationen ist das Medium des Response-Bodys, welches durch einen entsprechenden MIME-Typ spezifiziert wird. Abhängig vom MIME-Typ entscheidet der Browser, was mit dem übermittelten Dokument geschehen soll. Ist das Dokument vom Typ text/html, wird der Inhalt im Browserfenster angezeigt. Kann der Nachrichten-Typ einer auf dem Browserrechner installierten Applikation zugeordnet werden (Plug-In), kann die Applikation aufgerufen und der Inhalt der Nachricht angezeigt werden. Der Umgang mit den im Response-Body enthaltenen Informationen hängen von den Einstellungen des Browsers ab. Weitere Einträge des Headers sind optional und dienen dem Browser beispielsweise zur Verwaltung seines Caches (Datum und Uhrzeit der letzten Änderung des Dokuments). Eine vollständige Dokumentation zu HTTP befindet sich in sogenannten RFC's (Request for Comments). Die Definitionen befinden sich in [RFC 1945](#) (für HTTP/1.0) und [RFC 2068](#) (für HTTP/1.1).

## Kontakt

### Web-Schnittstelle - Kommunikation mit der Server-Erweiterung

#### Kommunikation mit der Datenbank

Wie in dem vorhergehenden Kapitel beschrieben, wird bei einem Request eine Verbindung zu einem Server mit einer bestimmten IP-Adresse aufgebaut. Im Request-Header wird dann die Quelle der angeforderten Daten angegeben. Normalerweise handelt es sich dabei um eine Datei, die vom Web-Server (IIS) an den Browser übermittelt wird.

#### Identifikation des Web-Servers

Ausschlaggebend für die Identifizierung des Web-Servers ist die IP-Adresse. Ein Web-Server kann über eine oder mehrere ("multi-homed") IP-Adressen angesprochen werden. Diese IP-Adressen können wiederum einer oder mehreren ("virtual") Sites zugeordnet sein. Jede dieser Sites kann über ein eigenes Verzeichnis verfügen, aus dem Dokumente bei einem Request übertragen werden. Besitzen mehrere Sites dieselbe IP-Adresse, so müssen diese durch unterschiedliche Hostnamen voneinander unterschieden werden.

Der IP-Adresse 62.132.119.100 ist beispielsweise der Site mit dem Hostnamen www.vectorsoft.de zugeordnet. Im IIS kann eine weitere Site definiert werden, die über die gleiche IP-Adresse angesprochen wird, aber mit dem Hostnamen shop.vectorsoft.de. Damit der Web-Server über den Hostnamen angesprochen werden kann, muss natürlich ein entsprechender Eintrag im DNS-Server erfolgen.

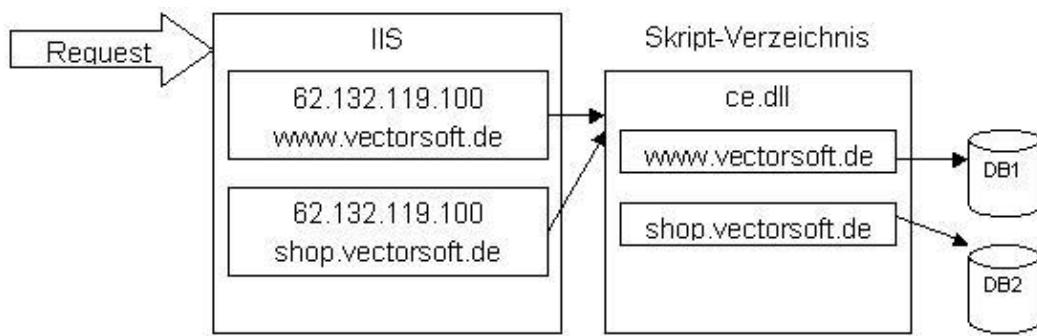
Die beiden Sites können jetzt nicht mehr alleine aufgrund ihrer IP-Nummer unterschieden werden. Die Unterscheidung wird dann anhand des Hostnamen getroffen. Der Browser trägt im Request-Header automatisch im Eintrag Host den Hostnamen ein. Nachdem die Site identifiziert ist, kann auf deren Ressourcen zugegriffen werden.

Aus der URL [http://shop.vectorsoft.de/scripts/c16\\_web.dll](http://shop.vectorsoft.de/scripts/c16_web.dll) wird über einen DNS die IP-Adresse ermittelt und eine Verbindung hergestellt. Der dazugehörige Request sieht wie folgt aus:

```
GET /scripts/c16_web.dll HTTP/1.1Host: shop.vectorsoft.de...
```

In den Ressourcen der Site shop.vectorsoft.de wird nach dem Verzeichnis /scripts gesucht und die Datei c16\_web.dll geladen. Im Unterschied zu anderen Dateien wird anhand der Dateierweiterung (.dll) erkannt, dass es sich nicht um eine Datei handelt, die an den Client übertragen werden soll. Es wird als Antwort also nicht die Datei c16\_web.dll selbst gesendet, vielmehr wird der gesamte Request an die Funktionen der DLL weitergeleitet.

## Kontakt



Es obliegt dem Client, den Request auszuwerten und eine Datei an den IIS zurückzugeben, die er dann an den Browser sendet. Die Ressourcen einer Site können sich mit den Ressourcen einer anderen Site überschneiden. So ist es möglich, dass zum Beispiel beide Sites ([www.vectorsoft.de](http://www.vectorsoft.de) und [shop.vectorsoft.de](http://shop.vectorsoft.de)) das gleiche Skript besitzen und damit denselben Client verwenden. Der Client wird so über die URL [http://www.vectorsoft.de/scripts/c16\\_web.dll](http://www.vectorsoft.de/scripts/c16_web.dll) und [http://shop.vectorsoft.de/scripts/c16\\_web.dll](http://shop.vectorsoft.de/scripts/c16_web.dll) angesprochen.

### Identifikation der Applikation

Mit einem Client können mehrere Applikationen (Datenbanken) betrieben werden. In einer Applikation werden Benutzeranfragen und Eingaben, die vom Browser übermittelt werden, ausgewertet und entschieden, welches Dokument an den Browser zurückgesendet wird. Die verschiedenen Applikationen werden in der Konfigurationsdatei des Clients eingetragen.

Die Identifikation der Applikation erfolgt über den Hostnamen bzw. die IP-Nummer. Benutzen mehrere Applikationen den selben Hostnamen, so muss die Anwendung zusätzlich über das Argument C16APP im Query-String identifiziert werden. Der im Request-Header eingetragene Hostname entspricht der Zeichenkette der URL, die zwischen dem // und dem ersten / steht. Dies kann sowohl ein Rechnername, als auch eine IP-Nummer sein. Einer einzelnen Applikation können auch mehrere Hostnamen bzw. IP-Nummern zugeordnet werden.

In der Konfigurationsdatei des Clients (Eintrag web\_url\_id) werden die Hostnamen angegeben, mit der die Applikation identifiziert wird. Um bei unserem Beispiel zwei unterschiedliche Applikationen zu erreichen, müssen zwei Einträge vorhanden sein. Für die erste Applikation muss web\_url\_id auf [www.vectorsoft.de](http://www.vectorsoft.de) und für die zweite Applikation in einem weiteren Abschnitt der Konfigurationsdatei auf [shop.vectorsoft.de](http://shop.vectorsoft.de) gesetzt werden (vgl. [Web-Schnittstelle - c16\\_web.cfg](#)).

Eine Applikation kann so aber nur dann angesprochen werden, wenn der Hostname in der URL angegeben wurde. Erfolgt der Aufruf des Clients über die URL [http://62.132.119.96/scripts/c16\\_web.dll](http://62.132.119.96/scripts/c16_web.dll), ist im Host-Eintrag nur die IP-Nummer vorhanden und eine Applikation kann nicht eindeutig ermittelt werden. Um eine Applikation sowohl unter dem Hostnamen, als auch unter der IP-Nummer zu erreichen, können zwei durch Semikolon getrennte Einträge in web\_url\_id vorgenommen werden:

```
web_url_id = www.vectorsoft.de;62.132.119.100
```

## Kontakt

Sollen mehrere Applikationen auch noch unter dem gleichen Hostnamen laufen, kann eine weitere Unterscheidung durch den Eintrag `web_app_id` getroffen werden. Der URL muss dann das Argument `C16APP` mit der Applikations-ID angehangen werden.

`http://www.vectorsoft.de/scripts/c16_web.dll?C16APP=Kunden`

In der Konfigurationsdatei wird dann `web_app_id = Kunden` angegeben.

## Kommunikation mit der Datenbank

Ist die Applikation identifiziert, wird ein Web-Benutzer angelegt. Dieser Web-Benutzer erhält eine eindeutige ID, bestehend aus einer 24stelligen Zeichenkette. Diese ID wird bei zukünftigen Anfragen an den Client im Query-String als Argument (`C16UID`) oder als Teil der URL übermittelt. Durch die Übertragung einer gültigen ID erfolgt auch automatisch die Zuordnung zur entsprechenden Applikation.

Nach dem Anlegen des Web-Benutzers baut der Client eine Verbindung zur Datenbank auf. Die dafür benötigten Informationen (Datenbank-Server, Datenbankname, Benutzername und Kennwort) werden in der Konfigurationsdatei des Clients im Abschnitt der Applikation eingetragen. Nach dem Öffnen der Datenbank existieren somit zwei "Benutzer" auf unterschiedlichen Ebenen. Zum einen gibt es einen Web-Benutzer, der im Client registriert ist, zum anderen gibt es einen Datenbank-Benutzer, der in der Datenbank angemeldet ist.

In der Datenbank wird anschließend die in der Konfigurationsdatei unter dieser Applikation angegebene Request-Funktion ausgeführt. Die Request-Funktion wertet die übergebene Browseranfrage aus und gibt ein Dokument an den IIS zurück. Das Dokument wird dann an den Browser zurückgesendet.

Handelt es sich bei dem Dokument um eine HTML-Seite, können dort wiederum Links mit Informationen für die Request-Funktion enthalten sein. Dieser Link verweist, wie in den obigen Beispielen, auf die `c16_web.dll` und übergibt im Query-String die entsprechenden Informationen. Diese Informationen werden wiederum von der Request-Funktion ausgewertet und mit einem Dokument beantwortet.

Das Dokument kann entweder vollständig aus der Datenbank generiert werden oder als externe Datei vorliegen. Es kann besondere Kommandos enthalten, die von dem Client vor dem Versenden interpretiert werden. Über diese Kommandos können Links in HTML-Seiten eingebunden oder weitere Funktionen in der Datenbank aufgerufen werden.

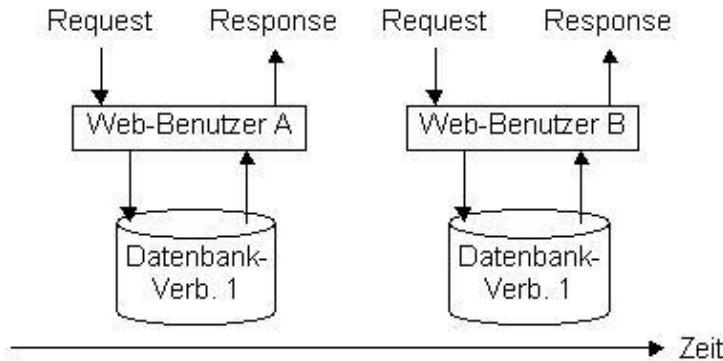
## Die Verbindung zwischen Client und Datenbank

Trifft ein Request für eine Applikation beim Client ein, wird zunächst überprüft, ob eine vom Client bereits eingerichtete Datenbankverbindung verwendet werden kann. Ist noch keine Datenbankverbindung vorhanden, wird ein entsprechender Benutzer bei der Datenbank angemeldet. Unter Verwendung der Datenbankverbindung wird dann wiederum die Request-Funktion aufgerufen.

Die Request-Funktion gibt ein Dokument zunächst an den Client zurück. Das entsprechende Dokument wird auf Befehle untersucht, die vom Client verarbeitet

## Kontakt

werden müssen. An dieser Stelle können weitere Funktionen aus der Datenbank aufgerufen werden. Diese Funktionen werden von dem gleichen Datenbankbenutzer ausgeführt. Nach der Rückgabe des so entstandenen Dokuments an den Web-Server (und damit an den Browser) ist der Request abgearbeitet. Der Web-Benutzer benötigt die Verbindung mit der Datenbank nicht mehr und gibt sie frei. Die Datenbankverbindung wird aber nicht sofort beendet. Trifft ein weiterer Request von einem beliebigen Web-Benutzer ein, steht die Datenbankverbindung jetzt für diesen Request zur Verfügung. Erst wenn die Datenbankverbindung über einen längeren Zeitraum nicht zur Bearbeitung eines Requests benötigt wird, wird der entsprechende Datenbankbenutzer abgemeldet und die Verbindung beendet.



Auf diese Weise kann eine Datenbankverbindung von mehreren Web-Benutzern verwendet werden. Werden von mehreren Web-Benutzern gleichzeitig Requests an den Client gesendet, können weitere Verbindungen zur Datenbank geöffnet werden.

Durch Einstellungen in der Konfigurationsdatei kann für jede Applikation die maximale Anzahl der Datenbankverbindungen individuell eingestellt werden (siehe Web-Schnittstelle - c16\_web.cfg).

Durch diese Verteilung der Web-Benutzer auf unterschiedliche Datenbank-Benutzer wird das Benutzerlimit des CONZEPT 16-Servers durch eine Web-Anwendung nicht über Gebühr beansprucht.

Dadurch, dass ein Datenbank-Benutzer von mehreren Web-Benutzern verwendet wird, ergeben sich einige Konsequenzen, die bei der Programmierung von Funktionen für die Web-Schnittstelle berücksichtigt werden müssen. Ein Request wird immer von einem Datenbank-Benutzer abgehandelt. Innerhalb eines Requests kann also wie gewohnt programmiert werden. Zustände bleiben aber nicht über einen Request hinaus erhalten. Lediglich, was in globalen Datenbereichen abgelegt wird, bleibt auch über den Request hinaus gespeichert.

### Beispiel:

In einem Request soll der nächste Datensatz einer Tabelle gelesen werden. Damit das richtige Ergebnis erzielt wird, muss im Request selbst oder in einer globalen Variablen der eindeutige Schlüsselwert des jetzigen Datensatzes abgelegt werden. Der Inhalt von Feldpuffern kann sich in der Zwischenzeit durch einen anderen Request (der den gleichen Datenbank-Benutzer verwendete) verändert haben. Davon sind ebenfalls dynamische Strukturen, benutzerdefinierte Feldpuffer, Textpuffer usw.

## Kontakt

betroffen.

**Von einem Web-Benutzer kann immer nur ein Request bearbeitet werden. Da während der Browser auf eine Antwort eines Requests wartet, weitere Requests an den Client gesendet werden können (z. B. durch wiederholtes Anklicken eines Links), wird ein weiterer Request des Web-Benutzers in einer Warteschlange abgelegt. Weitere Requests des Benutzers werden abgelehnt. Die Übertragung des Ergebnisses eines Request (z. B. Download einer Datei) kann aber auf einen Zeitpunkt verschoben werden, nachdem der Request abgearbeitet ist. Somit steht der Web-Benutzer wieder zur Verfügung, auch wenn die Datei, die zurückgesendet werden soll, noch übertragen wird.**

**Wird eine Seite an den Browser zurückgegeben, kann diese wiederum Links mit entsprechenden Parametern enthalten, die von der Request-Funktion ausgewertet werden können. Die Seite kann entweder vollständig aus der Datenbank generiert werden, oder es wird der Dateiname einer externen Datei angegeben. Die zurückgegebene Seite kann wiederum Prozeduraufälle in die Datenbank enthalten. Der Rückgabewert des Aufrufs wird in die Seite integriert. Eine HTML-Seite kann also auf zwei Wegen mit Datenbankinhalten gefüllt werden:**

**Die Seite wird in der Datenbank aufbereitet**

**Es wird eine unvollständige Seite zurückgegeben, die durch weitere Prozeduraufälle mit Informationen gefüllt wird**

**Trifft also ein Request ein, wird eine Prozedur durchgeführt, die eine HTML-Seite zurückgibt. Die Seite kann entweder Zeilenweise aus der Applikation generiert werden oder als Datei auf der Festplatte vorliegen. Die zurückgegebenen Zeilen oder die zu übertragende HTML-Seite kann neben HTML-Anweisungen C16.-Anweisungen enthalten. Diese Anweisungen werden durch die Web-Schnittstelle ausgewertet. Eine solche Anweisung kann zum Beispiel eine Funktion in der Datenbank aufrufen, die weitere Inhalte in die HTML-Seite integriert.**

**Neben den eigentlichen Daten werden Links in die Seite integriert, die ein Navigieren in der Applikation ermöglichen. Diese Links sollten so aufgebaut werden, dass aus ihnen die Seite, auf welcher der Link steht, hervorgeht. Der Browser hat immer die Möglichkeit eine vorhergehende Seite anzuzeigen, ohne das ein Request gesendet, also der Applikation der Seitenwechsel mitgeteilt wird. Anweisungen zum Erstellen dieser Links können in die HTML-Seiten integriert werden. Diese Anweisungen werden dann vom Client durch entsprechende Informationen ersetzt.**

## Kontakt

### Installation der Web-Schnittstelle

#### Beschreibung der Installation der CONZEPT 16-Web-Schnittstelle

Die Dateien der Web-Schnittstelle werden über die CONZEPT 16 Installationsroutine eingerichtet.  
Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \Web.

#### Dateien der Web-Schnittstelle

c16\_web\_iis7\_w32.dll Client-DLL für IIS7 32-Bit (\Inetpub\scripts)

c16\_web\_iis7\_w64.dll Client-DLL für IIS7 64-Bit (\Inetpub\scripts)

\example

Beispielkonfigurationsdateien der Web-Schnittstelle

c16\_web.ini

Initialisierungsdatei (\Inetpub\scripts oder \WinNT\System32)

c16\_web.cfg

Konfigurationsdatei (\Inetpub\scripts oder beliebiges Verzeichnis)

Die Datei c16\_web\_iis\*.dll ist die eigentliche Web-Schnittstelle. Damit der Server von einem Browser angesprochen werden kann, muss die Schnittstelle in ein Verzeichnis der Website kopiert werden, das über Script-Berechtigung verfügt. In der Standardinstallation ist dies das Verzeichnis \InetPub\scripts. Es muss nur eine der Dateien kopiert werden. Welche der Dateien hängt vom verwendeten Internet Informations Service ab:

Betriebssystem

zu kopierende Datei

Windows 7 (32-Bit) oder höher / IIS Version 7

c16\_web\_iis7\_w32.dll

Windows 7 (64-Bit), Windows Server 2008 R2 oder höher / IIS Version 7

c16\_web\_iis7\_w64.dll

Die Datei sollte nach c16\_web.DLL umbenannt werden. Es kann aber auch ein beliebiger anderer Name verwendet werden. Dadurch ändern sich auch die Namen aller anderen Dateien, da der Name bis auf die Dateiendung gleich sein muss.



Die CONZEPT 16-Web-Schnittstelle kann nur zusammen mit den Internet Information Services (ab Version 7) eingesetzt werden.

Die Konfiguration der Web-Schnittstelle wird über Konfigurationsdateien vorgenommen. Nachdem der CONZEPT 16-Server gestartet und die entsprechenden Einstellungen der Web-Schnittstelle getroffen wurden, kann mit einem Browser über die Web-Schnittstelle eine Datenbank angesprochen werden.

## Kontakt

### Web-Schnittstelle - Einstellungen des IIS

#### Notwendige Einstellungen bei den Internet Information Services Installation

Siehe der  
Web-Schnittstelle,

#### Einstellungen der c16\_web.dll

Die notwendigen Einstellungen sind von der Version des Internet Information Services abhängig.

##### Einstellungen des IIS Version 5 oder kleiner

##### Einstellungen des IIS Version 6

##### Einstellungen des IIS Version 7 / 8

#### Einstellungen des IIS Version 5 und kleiner

Beim Zugriff auf Inhalte des Internet-Servers spielen zwei Benutzerkonten eine Rolle. Das Benutzerkonto IUSR\_<Computername> wird verwendet, wenn ein Benutzer von einem Browser aus ohne authentifizierten Benutzer zugreift. Die CONZEPT 16-Web-Schnittstelle (c16\_web.dll) wird vom IIS unter Verwendung des Benutzerkontos IWAM\_<Computername> gestartet. <Computername> wird dabei durch den NetBIOS-Namen des Internet-Servers ersetzt. Beide Benutzer benötigen Schreibrechte im Verzeichnis der Protokolldateien (logfile und web\_log\_path).

Zum Betreiben des Clients müssen bis zur Version 6.0 des IIS keine weiteren Einstellungen beim IIS vorgenommen werden. Zum sicheren und schnellen Betrieb sollten aber folgende Einstellungen in den Eigenschaften des Scripts-Verzeichnisses berücksichtigt werden:

#### Getrennter Speicherbereich

Das Skript läuft in einem vom IIS getrennten Speicherbereich. Bei Problemen in der Verarbeitung von Prozeduren im Client muss nicht der komplette IIS neu gestartet werden. In der Regel genügt dann das Entladen des Skriptes.

#### Cache ISAPI-Applikationen

Über die Schaltfläche [Konfiguration...] können bestimmte Erweiterungen einem Skript zugeordnet werden. Die Web-Schnittstelle braucht hier nicht separat eingetragen werden. Auf dieser Seite muss aber zusätzlich angegeben werden, dass das Skript vom IIS im Speicher verbleibt, auch wenn es nicht verwendet wird ("Cache ISAPI Applikationen"). Dies ist zum Betrieb des Clients unbedingt notwendig.

#### Verfallszeitpunkt der zurückgegebenen Seiten

Beim IIS kann ein Verfallsdatum für HTML-Seiten eingetragen werden. Ein Verfallsdatum verhindert, dass diese Seiten auf einem Proxyserver über dieses Datum hinaus gespeichert werden. Da die von der Web-Schnittstelle kommenden Seiten immer wieder neu mit den aktuellen Inhalten aus der Datenbank generiert werden, erfolgt die Eintragung eines bereits vergangenen Verfallsdatums automatisch durch den Client. Besondere Einstellungen in diesem Bereich dürfen nicht erfolgen.

## Kontakt

### Einstellungen des IIS Version 6

Beim Einsatz der Version 6.0 des IIS sind folgende Einstellungen vorzunehmen:

#### Isolationsmodus

Die Option "WWW-Dienst im IIS 5.0-Isolationsmodus ausführen" muss gesetzt sein. Diese Einstellung befindet sich unter "Eigenschaften von Websites" - Registerkarte "Dienst".

#### Webdiensterweiterung

Die Web-Schnittstelle muss als Webdiensterweiterung registriert werden. Die Registrierung wird im Knoten "Webdiensterweiterungen" über "Neue Webdiensterweiterung hinzufügen" vorgenommen. Der Erweiterungsstatus muss auf "Zugelassen" gesetzt werden.

#### Anwendungsschutz

Bei den Anwendungseinstellungen wird empfohlen den "Anwendungsschutz" auf "Hoch (isoliert)" einzustellen.



Die Einstellung "Anwendungsschutz" steht nur dann zur Verfügung, wenn der "IIS 5.0-Isolationsmodus" gesetzt ist.

### Einstellungen des IIS Version 7 / 8

Da unter dem IIS 7 / 8 die CONZEPT 16 Web-Schnittstelle als systemeigenes Modul eingetragen werden kann, erfolgt die Konfiguration relativ einfach. Standardmäßig wird der Benutzer "NetworkService" für die Ausführung innerhalb der Anwendungspools verwendet. Dieser Benutzer benötigt entsprechende Rechte zum Ausführen der Schnittstelle und zum Schreiben der Protokolldateien. Folgende Einstellungen müssen vorgenommen werden:

#### Registrieren der Web-Schnittstelle

Zunächst muss die DLL der Web-Schnittstelle bei den Internet Information Services registriert werden. Dazu wird der Root-Knoten (das Webserver-System) und anschließend der Punkt "Module" ausgewählt. Hier kann über die Funktion "Systemeigenes Modul konfigurieren..." eine neuen Modul registriert werden. Über die Schaltfläche "Registrieren..." wird die Web-Schnittstelle eingetragen. Dabei werden ein frei wählbarer Name und der Pfad angegeben. Falls die Web-Schnittstelle nicht für alle eingetragenen Websites aktiviert werden soll, muss hier das aktivierte Häkchen entfernt werden.

#### Aktivieren der Schnittstelle für eine Website

Um die Web-Schnittstelle für eine Website zu aktivieren wird die Website ausgewählt und anschließend der Punkt "Module" geöffnet. Hier taucht die bereits registrierte Schnittstelle auf und kann über das Aktivieren des Kontrollhäckchens gestartet werden.

#### Erneutes Laden der Konfigurationseinstellungen

## **Kontakt**

**Um die Konfigurationseinstellungen der Web-Schnittstelle erneut zu laden, wird der Anwendungspool zur entsprechenden Website neu gestartet. Dazu kann über den Bereich "Anwendungspools" der betroffene Pool ausgewählt und mit der Funktion "Wiederverwenden" gestoppt und wieder gestartet werden.**

### **Update der Schnittstelle**

**Um die DLL der Schnittstelle zu aktualisieren, wird wie beim erneuten Laden der Konfiguration der Anwendungspool der Website gestoppt (Funktion "Beenden"). Nach dem Update kann er dann wieder gestartet werden (Funktion "Starten").**

## Kontakt

### Web-Schnittstelle - Einstellungen der c16\_web\_iis\*.dll

#### Konfiguration der Web-Schnittstelle

Einstellungen  
des Internet  
Information

Siehe

Services,  
Einstellungen

Alle Einstellungen der Web-Schnittstelle erfolgen in der Initialisierungs- und der Konfigurationsdatei. Die Initialisierungsdatei (c16\_web.ini) kann in das Verzeichnis der Web-Schnittstelle (c16\_web.dll) oder in das System32-Verzeichnis von Windows kopiert werden.

Name und Pfad der Konfigurationsdatei c16\_web.cfg können in der Initialisierungsdatei angegeben werden. Ist ein entsprechender Eintrag nicht vorhanden, muss die Konfigurationsdatei im gleichen Verzeichnis stehen wie die Web-Schnittstelle und den Namen c16\_web.cfg haben. Beide Dateien liegen im ASCII-Format vor und können mit einem einfachen Editor bearbeitet werden.

Kommentare können mit einem ; eingeleitet werden. Der Rest der Zeile wird dann nicht ausgewertet.

## Kontakt

### Web-Schnittstelle - c16\_web.ini

#### Inhalt der Datei c16\_web.ini

Siehe Einstellungen

der

c16\_web.dll,

Diese Datei muss im gleichen Verzeichnis wie die Datei c16\_web\_iss\*.dll oder im System32-Verzeichnis von Windows stehen. Über folgenden Link finden Sie eine Beispiel-Datei.

Einträge, die mit einem \* markiert sind, müssen in der Datei eingetragen sein.

Folgende Einstellungen können in dieser Datei vorgenommen werden:

max\_memory\_mb  
configfile  
logfile  
serverv5

**max\_memory\_mb\***

In diesem Eintrag wird die maximale Speichermenge in Megabyte angegeben, die vom Client reserviert werden kann. Je nach Betriebssystem können 16 MB bis maximal 1,5 bis 2 GB Speicher angefordert werden. Diese Angabe stellt nur eine obere Grenze dar, die nicht überschritten werden darf.

**max\_memory\_mb = 128**

**configfile\***

Dieser Parameter muss nur dann angegeben werden, wenn sich die Konfigurationsdatei nicht im gleichen Verzeichnis befindet oder einen anderen Namen als c16\_web.CFG besitzt. In diesem Eintrag wird der Pfad und der Name der Konfigurationsdatei angegeben. Die Erweiterung der Datei ist immer .CFG und wird nicht mit angegeben.

**configfile = c:\c16\web\c16\_web**

Aus Sicherheitsgründen sollte sich die Konfigurationsdatei nicht im gleichen Verzeichnis wie die c16\_web\_iss\*.dll befinden.

**logfile\***

Im Logfile werden Ereignisse beim Betrieb des Clients protokolliert. Für jeden Tag wird eine eigene Datei angelegt. Ist dieser Eintrag nicht definiert, wird im gleichen Verzeichnis, in dem auch die Datei c16\_web.DLL abgelegt ist, eine Datei mit dem Namen c16\_web.<Datum>.LOG angelegt. Das Datum wird dabei ohne Trennzeichen in der Form JJJJMMTT angegeben. In diesem Eintrag kann ein Verzeichnis und ein Dateiname für die Logdatei angegeben werden. Die Erweiterung der Datei ist immer .LOG und wird nicht mit angegeben.

**logfile = c:\c16\web\c16\_web**

## Kontakt

Aus Sicherheitsgründen sollte sich die Protokolldatei nicht im gleichen Verzeichnis wie die c16\_web\_iss\*.dll befinden.

Mit diesem Eintrag werden zum Beispiel vom 31. Mai 2008 bis zum 2. Juni 2008 folgende Dateien angelegt:

c:\c16\web\c16\_web.20080531.log;c:\c16\web\c16\_web.20080601.log;c:\c16\web\c16\_web.20080602.log

serverv5

Diese Eintragung wird nur benötigt, wenn mit einer älteren Version (2.2.14) auf einen CONZEPT 16-Datenbankserver der Version 5.0 oder höher zugegriffen werden soll.

serverv5 = 1



In den Einstellungen configfile und logfile muss darauf geachtet werden, dass der IIS (und somit auch der Client) nur lokale Laufwerke und Verzeichnisse ansprechen kann, da der Web-Server als Dienst gestartet wird.

## Kontakt

### Web-Schnittstelle - c16\_web.cfg

#### Inhalt der Datei c16\_web.cfg

c16\_web.INI,  
Einstellungen

Siehe der  
c16\_web.DLL

In der Datei steht am Anfang ein allgemeiner Bereich, in dem Informationen für den gesamten Client abgelegt werden. Für jede Applikation wird danach ein eigener Bereich eingetragen. Einträge die mit einem \* markiert sind, müssen in der Datei vorhanden sein. Eine mögliche Konfiguration befindet sich in der Beispiel-Datei.

Einträge im allgemeinen Bereich

web\_max\_connections\*  
web\_max\_sessions\*  
web\_max\_timeout\_s\*  
web\_min\_timeout\_s\*  
web\_req\_timeout\_s

Einträge im Applikationsbereich

web\_root\_path\*  
web\_error\_path\*  
web\_log\_path  
web\_module\_path\*  
web\_url\_id\*  
web\_app\_id  
web\_uid\_mode  
c16\_server\*  
c16\_database\*  
c16\_user\*  
c16\_password\*  
c16\_proc\_request\*  
c16\_proc\_cache\_kb  
c16\_proc\_test  
c16\_max\_connections\*  
c16\_min\_delay\_ms\*  
c16\_max\_delay\_ms\*  
c16\_timeout\_s\*

Einträge im allgemeinen Bereich

web\_max\_connections\*

Die maximale Anzahl aller Web-Benutzer des Clients können in diesem Eintrag angegeben werden. Will sich ein weiterer Benutzer anmelden, wird zunächst versucht, einen vorhandenen Web-Benutzer abzumelden, der über einen längeren Zeitraum keine Anfragen an den Client gestellt hat. Welcher Benutzer entfernt wird, ist beim Eintrag web\_min\_timeout\_s beschrieben. Kann kein Web-Benutzer entfernt werden, wird ein entsprechender Fehler an den Browser zurückgeliefert. Dieses Limit ist nicht applikationsspezifisch und dient der Beschränkung des Verbrauchs an

## Kontakt

Systemressourcen durch den Client.

In dieser Einstellung können Werte von 5 bis 100000 angegeben werden.

**web\_max\_sessions\***

In diesem Eintrag wird festgelegt, wie viele Sessions von einer einzelnen Browser-IP-Adresse aus gleichzeitig möglich sind. Auf diese Weise kann nur eine beschränkte Anzahl von Browser-Sessions von einem Rechner aus auf den Client zugreifen.

Diese Beschränkung besteht ebenfalls, wenn über einen Proxy-Server auf den Client zugegriffen wird. Alle Anwender aus dem Netz, das durch den Proxy vertreten wird, melden sich mit der gleichen IP-Adresse (die des Proxy) beim Client an. Damit wird dann die Anzahl der Sessions aus einem Netz beschränkt.

Wird web\_max\_sessions mit 0 angegeben, findet keine Beschränkung der Sessions statt. Es können maximal 10000 Sessions zugelassen werden. Eine Einschränkung ist sinnvoll, um Denial-of-service-Attacken mittels Session-Overflow zu verhindern. Ein empfohlener Wert ist 20.

**web\_max\_timeout\_s\***

Ein Web-Benutzer wird vom Client entfernt, wenn er nach der hier eingestellten Anzahl von Sekunden keine Anfrage mehr gestellt hat. Ist web\_max\_timeout\_s auf 900 Sekunden gesetzt, wird ein Web-Benutzer, der 15 Minuten keine Anfrage mehr gestellt hat, vom Client abgemeldet. Die ID des Web-Benutzers ist danach nicht mehr gültig.

Die Einstellung kann zwischen 120 und 10800 Sekunden (oder zwischen 2 Minuten und 3 Stunden) gewählt werden. Der Wert sollte kleiner sein als der Wert, der in der Einstellung c16\_timeout\_s angegeben ist.

 Bei der Angabe von Zeiten können entweder die Anzahl der Sekunden angegeben werden oder eine Zeitangabe in der Form [hh.]mm.ss.

**web\_min\_timeout\_s\***

Hier wird eingestellt, wie viele Sekunden ein Web-Benutzer mindestens keine Anfrage mehr an den Client gestellt hat, um bei Bedarf entfernt werden zu können. Diese Einstellung wird erst dann benötigt, wenn die in web\_max\_connections eingestellte Anzahl von Web-Benutzern erreicht ist und sich ein weiterer Web-Benutzer anmelden möchte.

Zu diesem Zeitpunkt wird ein Web-Benutzer gesucht, der über einen Zeitraum von mehr als den hier eingestellten Sekunden keine Anfrage mehr gestellt hat. Werden mehrere Web-Benutzer gefunden, die diese Bedingung erfüllen, wird der Web-Benutzer entfernt, der am längsten keine Anfrage mehr gestellt hat.

**Beispiel:**

## Kontakt

Im Client sind 200 Web-Benutzer registriert, was genau der Eintragung in web\_max\_connections entspricht. 198 dieser Benutzer stellen in kurzen Zeitabständen (weniger als web\_min\_timeout\_s Sekunden) Anforderungen an die Datenbank. Der Web-Benutzer A hat seit 600 Sekunden, der Web-Benutzer B seit 450 Sekunden keine Anforderung gestellt. Soll jetzt ein weiterer Web-Benutzer angemeldet werden, wird der Web-Benutzer A entfernt, sofern web\_min\_timeout\_s kleiner als 600 Sekunden ist.

Die Einstellung kann zwischen 120 und 1800 Sekunden (oder zwischen 2.00 und 30.00) gewählt werden.

### web\_req\_timeout\_s

Es wird immer nur ein Request eines Web-Benutzers auf einmal verarbeitet. Da der Browser asynchron zum Web-Server arbeitet, können von einem Browser mehrere Requests an den Server gesendet werden, bevor die Antwort auf den ersten Request zurückgeliefert worden ist. Beispielsweise kann während der Bearbeitung eines Requests der Benutzer in seinem Browser einen weiteren Link anklicken. Dieser zusätzliche Request wird im Client in einen Wartezustand versetzt und erst dann verarbeitet, wenn der erste Request beendet wurde. Jeder weitere Request ersetzt den Request im Wartezustand. Es steht also höchstens ein weiterer Request in der Warteschlange. Der Request wird aus der Warteschlange entfernt, wenn er innerhalb der angegebenen Zeit (in Sekunden) nicht bearbeitet werden kann.

Es kann eine Wartezeit von 5 bis 600 Sekunden eingestellt werden. Befindet sich kein Eintrag in der Konfigurationsdatei, werden 10 Sekunden angenommen.

### Einträge im Applikationsbereich



Ein Applikationsbereich wird in der Konfigurationsdatei mit dem Namen der Applikation in eckigen Klammern [...] eingeleitet.

#### web\_root\_path\*

Hier wird ein Verzeichnis auf der Festplatte des Web-Servers angegeben. Das Verzeichnis kann über den Prozedurbefehl WseInfo( WseInfoRootPath) ermittelt werden. Alle Angaben in der Applikation mit Bezug auf eine externe Datei können relativ zu diesem Eintrag definiert werden. Durch dieses Vorgehen kann eine leichtere Übertragbarkeit auf andere Serverrechner erreicht werden. Es können bis zu 127 Zeichen angegeben werden.

```
web_root_path = c:\inetpub\wwwroot
```

#### web\_error\_path\*

Tritt ein Fehler auf, der nicht von der Request-Funktion verarbeitet werden kann, wird eine HTML-Seite aus dem hier angegebenen Verzeichnis angezeigt. Das Verzeichnis kann bis zu 127 Zeichen lang sein.

```
web_error_path = c:\inetpub\wwwroot\error
```

Der Aufruf von Fehlerseiten ist im Abschnitt Rückgabe von Fehlerseiten beschrieben.

## Kontakt

### **web\_log\_path**

In der Log-Datei werden Ereignisse der Applikation protokolliert. Die Datei wird in dem hier angegebenen Verzeichnis angelegt. Es können maximal 127 Zeichen angegeben werden. Für jeden Tag wird eine eigene Datei generiert. Ist diese Eigenschaft nicht gesetzt, wird im Verzeichnis, in dem auch die Datei c16\_web.DLL abgelegt ist, eine Datei mit dem Namen <Applikation>.<Datum>.LOG angelegt. Der Applikationsname ist in eckigen Klammern [...] in der Konfigurationsdatei angegeben. Das Datum wird dabei im Format JJJJMMTT angegeben.

```
web_log_path = c:\c16\web
```

### **web\_module\_path\***

In diesem Eintrag werden das Verzeichnis und der Name des Clients angegeben. Die Angabe ist relativ zu web\_root\_path. Das Verzeichnis kann über den Prozedurbefehl WseInfo( WseInfoModulePath) ermittelt und zusammen mit der Eintragung in web\_root\_path zur Generierung von URL's verwendet werden.

```
web_module_path = /scripts/c16_web.dll
```

### **web\_url\_id\***

In diesem Eintrag wird ein Hostname angegeben, mit dem diese Applikation identifiziert werden kann. Hier muss der Hostname der Site angegeben werden, unter dem die Applikation erreichbar ist. Der Hostname wird im Header jedes Browser-Request übergeben. Dabei kann es sich um einen Namen oder eine IP-Adresse handeln. Mehrere Namen können durch ; voneinander getrennt angegeben werden, damit kann die Applikation dann durch verschiedene Hostnamen angesprochen werden. Es können maximal 255 Zeichen angegeben werden.

Ist beim Web-Server ein anderer Port als der Standardport (80) angegeben, kann im ersten Eintrag die Portnummer mit angegeben werden.

```
web_url_id = shop.vectorsoft.de:8080;appl;62.132.119.100
```

### **web\_app\_id**

Sind mehrere Applikationen unter demselben Hostnamen erreichbar, wird über diese Applikations-ID die Applikation identifiziert. Die ID kann bis zu 30 Zeichen lang sein. Der hier eingetragene Wert muss im Query-String als Argument übergeben werden:

```
http://www.vectorsoft.de/scripts/c16_web.dll?C16APP=kunden
```

Der Name des Rechners wird der Einstellung web\_url\_id, der Pfad des Clients wird dem Eintrag web\_module\_path entnommen.

Erhält der Client eine URL in dieser Form, wird aufgrund von C16APP die Applikation identifiziert und deren Request-Funktion (c16\_proc\_request) aufgerufen. Beim ersten Aufruf können bereits weitere Argumente übergeben werden, die von der Request-Funktion ausgewertet werden können. Die Angabe von C16APP ist nur dann

## Kontakt

erforderlich, wenn noch keine Benutzer-ID verfügbar ist.

### **web\_uid\_mode**

**Die Benutzer-ID des Web-Benutzers wird grundsätzlich in der URL mit angegeben (Cookies werden nicht verwendet). Dabei wird die ID normalerweise in den Pfad integriert:**

`http://www.vectorsoft.de/scripts/c16_web.dll/C16UID....?....`

Mit der Angabe von `web_uid_mode = query` wird die ID dagegen in den Query-String übernommen:

`http://www.vectorsoft.de/scripts/c16_web.dll?C16UID=...&....`

**Die Angabe von `web_uid_mode = path` entspricht der Standardeinstellung. Die Pfadmethode ist vorzuziehen, da sonst die Benutzer-ID immer ein Argument des Query-Strings darstellt und dies bei Argumentabfragen mittels WseArg() berücksichtigt werden muss.**



**Die Angabe der Web-Benutzer-ID im Pfad ist nur ab IIS Version 4.0 möglich. Bei der Verwendung des IIS Version 3.0 muss `web_uid_mode` auf `query` gesetzt werden.**

### **c16\_server\***

In diesem Eintrag werden das Protokoll und der Name des Servers angegeben, auf dem die Anwendungs-Datenbank geöffnet werden soll. Die Angabe erfolgt nach dem Schema TCP:<Name des Servers>. "TCP" steht dabei für das unterstützte Protokoll TCP/IP.

Als Name des Servers kann entweder die IP-Adresse des Rechners, auf dem der CONZEPT 16-Server läuft, oder dessen Name, angegeben werden. Bei der Verwendung des Namens muss ein Domain Name Service im System installiert und der entsprechende Name dort eingetragen sein. Wird die Datenbank im Hot-Standby betrieben, werden die IP-Adressen der beiden Server durch "+" getrennt angegeben.

`c16_server = TCP:10.0.0.1+10.0.0.2`

Es können maximal 60 Zeichen angegeben werden.

### **c16\_database\***

Die zu öffnende Datenbank wird in diesem Eintrag angegeben. Die Datenbank muss so angegeben werden, wie sie beim Server definiert ist, d. h. ein entsprechender Eintrag muss in der Datenraumtabelle des Servers vorhanden sein. In diesem Eintrag können bis zu 127 Zeichen angegeben werden.

`c16_database = ecm`

### **C16\_user\* und C16\_password\***

## Kontakt

Die Anmeldung in der Datenbank erfolgt über den hier angegebenen Benutzer. Alle Web-Benutzer der Applikation melden sich in die angegebene Datenbank unter diesem Benutzernamen an. Ist zur Anmeldung ein Kennwort erforderlich, wird dieses im Eintrag c16\_password angegeben.

c16\_user = webuserc16\_password = mausi

Die Angabe des leeren Kennwertes erfolgt mit "". Benutzer und Kennwort können mit je maximal 20 Zeichen angegeben werden.

c16\_proc\_request\*

Die Kommunikation zwischen Web-Browser und Web-Server erfolgt über Anforderungen (Requests) des Browsers. Alle Requests an die Applikation werden von der hier angegebenen Prozedur bzw. Funktion verarbeitet. Die Angabe erfolgt über <Prozedur>:<Funktion>. Wird kein Funktionsname angegeben, wird die Main-Funktion der Prozedur aufgerufen. Der Eintrag darf 61 Zeichen nicht überschreiten. Wird kein Funktionsname angegeben, darf der Name nicht länger als 20 Zeichen sein.

c16\_proc\_cache\_kb

In dem eingestellten Cache werden Prozeduren zwischengespeichert. Der Cache wird für jede Datenbankverbindung eingerichtet. Der Prozedurcache sollte mindestens so groß wie die Request-Funktion ausfallen, da diese bei jeder Browser-Anfrage aufgerufen wird. Der Prozedurcache kann zwischen 32 und 65536 KB (64 MB) eingestellt werden. Ist dieser Eintrag nicht in der Konfigurationsdatei enthalten, wird ein Prozedurcache von 256 KB eingerichtet.

c16\_proc\_test

Der Prozedurcache einer Datenbankverbindung bleibt bis zum Beenden der Verbindung erhalten. Dies führt während der Entwicklungsphase dazu, dass Prozeduränderungen sich nicht auswirken, da sich die vorhergehende Version des Prozedurcodes noch im Cache befindet. Mit der Einstellung c16\_proc\_test = Y wird der Prozedurcache nach jedem verarbeiteten Request geleert, ohne dass die Datei c16\_web.DLL geladen werden muss. Diese Einstellung sollte im Echtbetrieb nicht aktiviert sein, da sie die Verarbeitung deutlich verlangsamt.

c16\_max\_connections\*

In dieser Eigenschaft wird die maximale Anzahl von Verbindungen zur Datenbank angegeben.

Die Einstellung unterscheidet sich von web\_max\_connections, in der die maximale Anzahl aller Web-Benutzer des Clients angegeben wird, während hier die maximale Anzahl der Verbindungen (Datenbankbenutzer) zur Datenbank für eine Applikation angegeben wird. Eine Verbindung wird nur solange genutzt, bis ein Request abgearbeitet wurde und steht danach einem anderen Web-Benutzer zur Verfügung. Der Eintrag kann zwischen 1 und 1000 gewählt werden.

## Kontakt

In diesem Eintrag dürfen nicht mehr Benutzer zugelassen werden, als der CONZEPT 16-Server ermöglicht. Der hier eingetragene Wert entspricht der Anzahl von Requests, die gleichzeitig bearbeitet werden können. Wird die Datenbank sowohl vom Web, als auch von eigenen Mitarbeitern genutzt, sollte die Anzahl der Verbindungen so eingestellt werden, dass auf jeden Fall alle Mitarbeiter arbeiten können.

### c16\_min\_delay\_ms\*

Wird ein Request an den Client gestellt und alle verfügbaren Datenbankverbindungen der Applikation sind bereits mit der Bearbeitung eines Requests beschäftigt, wird nach Ablauf der hier eingetragenen Millisekunden eine neue Datenbankverbindung aufgebaut. Wird in diesem Zeitraum eine Datenbankverbindung frei, wird diese zur Bearbeitung des Requests verwendet.

Hier können Werte von 0 bis 3000 eingetragen werden. Bei 0 wird sofort eine neue Verbindung eingerichtet.

### c16\_max\_delay\_ms\*

Kann keine weitere Datenbankverbindung eingerichtet werden, wird maximal die hier eingestellte Zeit gewartet. Wird eine Datenbankverbindung in diesem Zeitraum frei, wird diese verwendet. Kann auch nach dieser Zeit keine Verbindung benutzt werden, wird eine Fehlerseite mit dem HTTP-Status 503 bzw. dem Fehlerwert 50026 (siehe [Rückgabe von Fehlerseiten](#)) zurückgegeben.

Hier können Werte zwischen 100 und 10000 eingetragen werden.

### c16\_timeout\_s\*

Wird eine Datenbankverbindung länger als die hier eingestellte Zeit nicht benutzt, wird sie automatisch beendet, d. h. der Datenbankbenutzer wird abgemeldet. Es können Werte von 5 bis 14400 Sekunden eingetragen werden.

Der eingestellte Wert sollte größer sein, als die Werte in den Eintragungen web\_max\_timeout\_s und web\_min\_timeout\_s.

Das Zusammenspiel der Einträge c16\_max\_connections, c16\_min\_delay\_ms, c16\_max\_delay\_ms und c16\_timeout\_s funktioniert wie folgt. Im Beispiel wird von folgenden Einstellungen ausgegangen:

`c16_max_connections = 10c16_min_delay_ms = 50c16_max_delay_ms = 2000c16_timeout_s`

Ein Anwender stellt über einen Browser einen Request an den Client. Da der Web-Benutzer der erste ist, besteht noch keine Verbindung zur Datenbank. Der Client öffnet die Datenbank mit dem eingestellten Benutzernamen. Während der Request des ersten Anwenders bearbeitet wird, trifft ein weiterer Request eines zweiten Anwenders ein. Da bereits eine Datenbankverbindung besteht, wird zunächst überprüft, ob die Verbindung noch verwendet wird. Das ist der Fall, da der Request von Anwender #1 noch nicht abgeschlossen ist. Der Client wartet jetzt maximal 50 Millisekunden (c16\_min\_delay\_ms). Sollte in dieser Zeit die Datenbankverbindung nicht mehr für den Anwender #1 benötigt werden, wird sie von Anwender #2

## Kontakt

verwendet. Anwender #2 ist dann mit der gleichen User-ID in der Datenbank eingeloggt, wie kurz zuvor noch Anwender #1.

Wird die Datenbankverbindung weiter von Anwender #1 benötigt, wird nach insgesamt 50 Millisekunden eine weitere Verbindung zur Datenbank hergestellt. Die Anfragen von Anwender #1 und Anwender #2 werden gleichzeitig verarbeitet.

Nachdem der Request von Anwender #1 abgearbeitet wurde, bleibt die Verbindung zunächst bestehen. Trifft innerhalb der nächsten 600 Sekunden (c16\_timeout\_s) eine weitere Anfrage ein, wird die erste Verbindung erneut verwendet. Nach 600 Sekunden ohne Anfrage wird die Datenbankverbindung beendet.

Sind nun 10 (c16\_max\_connections) Datenbankverbindungen aufgebaut, von denen alle verwendet werden und es trifft ein weiterer Request ein, wird zunächst 50 Millisekunden gewartet. Wenn in dieser Zeit keine der Verbindungen frei wird, versucht der Client eine weitere Datenbankverbindung einzurichten. Dies ist nicht möglich, da die maximale Anzahl an Verbindungen bereits erreicht ist. Der Client versucht dann weitere 2000 Millisekunden (c16\_max\_delay\_ms) eine freie Verbindung zu erhalten. Wird keine der Verbindungen frei, wird der Request abgewiesen und der Anwender erhält eine entsprechende Fehlerseite zurück.

Durch diese Form des Verbindungs-Sharings können mehrere Web-Benutzer eine Datenbankverbindung nutzen und belasten damit nicht das Benutzerlimit des Servers. Es ist allerdings darauf zu achten, dass in der Programmierung nicht die User-ID UserInfo( UserCurrent) der Datenbank über einen Request hinaus verwendet wird, da diese von der Verbindung abgeleitet wird und sich daher für den Web-Benutzer bei jedem Request ändern kann. Stattdessen kann über den Befehl WseInfo( WseInfoUserNumber) die Nummer des Web-Benutzers ermittelt werden.

## Kontakt

**Web-Schnittstelle - Einstellungen des Browsers**

**Konfiguration des Internet-Browsers**

Einstellungen  
der

Siehe c16\_web.dll,  
Aufruf von  
Prozeduren

Beim Browser müssen keine besonderen Einstellungen vorgenommen werden. Es werden weder Plug-In's noch Cookies zur Kommunikation benötigt.

Das Layout der erstellten Seiten sollte in jedem Fall mit unterschiedlichen Browsern getestet werden. Es gibt einige Unterschiede in der Darstellung der Seiten. Der Microsoft Internet Explorer verwendet beispielsweise für die Beschriftung von Schaltflächen eine andere Schrift als der Netscape Communicator. Entscheidender sind Unterschiede in der Darstellung des Hintergrundes in Tabellen-Zellen und den Randeinstellungen.

Eine erstellte Seite sollte mit den gängigsten Browsern auf Darstellungsabweichungen untersucht werden, um ungewollte Effekte zu vermeiden.

## Kontakt

### Web-Schnittstelle - Aufruf von Prozeduren Aufruf von Prozeduren von der Web-Schnittstelle

Siehe [Einstellungen des Browsers](#),

#### [Anweisungen im HTML-Dokument](#)

Die Verarbeitung von Benutzereingaben der Web-Anbindung erfolgt über die in der Konfigurationsdatei eingetragene Request-Funktion ([c16\\_web.cfg](#)) und über Prozeduren, die aus einer HTML-Seite mit dem Befehl [C16.CALL\(\)](#) aufgerufen werden. Zur Programmierung der entsprechenden Funktionen stehen die Prozedurbefehle des CONZEPT 16-Befehlssatzes zur Verfügung.



**Von der Web-Schnittstelle können nur A+ Prozeduren verarbeitet werden. Der Aufruf von Prozeduren mit dem Befehl [CallOld\(\)](#) wird ignoriert. Innerhalb der Prozeduren stehen nicht alle Befehle zur Verfügung. Alle verwendbaren Befehle sind in der Online-Hilfe mit dem Symbol versehen.**

Die eingetragene Request-Funktion nimmt alle Anfragen eines Browsers entgegen. In dieser Funktion sollte aufgrund der vom Browser übermittelten Daten eine neue Seite zurückgegeben werden. Der URL der Applikation (bei einer Standardinstallation [http://<Hostname>/scripts/c16\\_web.dll](http://<Hostname>/scripts/c16_web.dll)) werden bei Verwendung des Clients unterschiedliche Argumente angehängt. Diese Argumente können mit dem Befehl [WseArg\(\)](#) in der Request-Funktion ermittelt und aufgrund des Ergebnisses unterschiedliche Seiten zurückgegeben werden.

In der Request-Funktion müssen unterschiedliche Zustände überprüft werden. Diese Zustände können über den Befehl [WseStatus\(\)](#) ermittelt werden. Die Request-Funktion besteht damit zunächst aus einer [switch-Anweisung](#):

```
switch (WseStatus()){  case _WseUserInit : {      // Request-Funktion zum ersten Mal aufgerufen
```

Im Zustand [WseUserInit](#) können globale Datenbereiche für die Applikation angelegt und initialisiert werden. In diesem Zustand wird die erste Seite der Applikation zurückgegeben. Die hier angelegten Datenbereiche können im Zustand [WseUserTerm](#) wieder freigegeben werden.

Der Zustand [WseUserTerm](#) wird nicht durch einen Request erreicht. Dadurch sind nur begrenzte Möglichkeiten vorhanden. So ist es zum Beispiel an dieser Stelle nicht möglich eine Antwortseite zurückzuschicken.

Der Zustand [WseUserHTML](#) ist nur gegeben, wenn die Funktion aus einer HTML-Seite (mittels [C16.CALL\(\)](#)) aufgerufen wurde. Wegen der Übersichtlichkeit sollten Funktionen, die Inhalte von Seiten erzeugen und Funktionen, die zur Ablaufsteuerung der Applikation dienen, voneinander getrennt werden. Der Zustand [WseUserHTML](#) kann dann in der Request-Funktion nicht vorkommen.

Im laufenden Betrieb wird bei einem Request die URL mit einer Benutzer-ID übermittelt. Durch den Request wird die Request-Prozedur aufgerufen. In der Prozedur wird dann der Status [WseUserProc](#) ermittelt. Der URL können ein oder mehrere Argumente angehängt werden, die in diesem Zustand ermittelt werden können. Je nach ermittelten Argumenten kann dann die Prozedur unterschiedliche Seiten zurückgeben. In diesem Bereich bietet sich ebenfalls eine [switch-Anweisung](#)

## Kontakt

an:

```
// Request-Funktion bei laufender Applikation aufgerufen  
aArgName # WseArg(_WseArgName,'',1); switch
```

In dem Beispiel wird der Name des ersten Arguments abgefragt. An dieser Stelle ist es günstiger den Namen über ein Namensfragment (z. B. App\*) zu ermitteln. Im einfachsten Fall wird dann mit dem Befehl WseReturn() eine Seite zurückgegeben.

```
WseReturn(_WseRetFile, 0, 'text/html', 'c:\inetpub\wwwroot\default.htm');
```

Eine Seite kann auch vollständig aus der Datenbank generiert und abschnittsweise zurückgegeben werden.

```
WseReturn(_WseRetString, 0, 'text/html', '<p>Weiter mit unten stehendem Link.</p>' + '<a href="
```

Der in dem Rückgabewert angegebene Link, kann dann wieder von der Request-Funktion ausgewertet werden. Als Name wird 'Button' und als Wert wird 'Weiter' ermittelt.



Kann eine Funktion nicht fehlerfrei ausgeführt werden, wird eine Fehlermeldung in die Protokolldatei eingetragen und gegebenenfalls eine Fehlerseite zurückgegeben.

Die Anweisung C16.URL() erzeugt einen Link auf die Web-Schnittstelle.

## Kontakt

Web-Schnittstelle - Anweisungen im HTML-Dokument

Übersicht der Befehle innerhalb einer HTML-Seite

Aufruf von  
Prozeduren,

Siehe

Befehle der  
Web-Schnittstelle

Nach dem Eingang eines Requests, wird die Request-Prozedur gestartet. In dieser Prozedur wird eine Datei entweder zeilenweise oder komplett als Response versendet. Bevor diese Datei durch den Web-Server zum Browser geschickt wird, durchsucht der Client die Seite nach den unten beschriebenen Anweisungen. Diese Befehle werden entweder durch einen Ergebniswert oder einen Leerstring ersetzt. Die HTML-Befehle der Seite bleiben davon unberührt. Im Gegensatz zu konventionellen Programmiersprachen werden sogar innerhalb eines Strings ("string") und innerhalb von Kommentaren (<!-- ... -->) die C16.-Anweisungen ausgewertet. Muss die Zeichenfolge C16. als einfacher Text eingesetzt werden (keine Anweisung), so ist sie durch & zu entwerten, also als &C16. einzugeben.

Das Durchsuchen der Seite kann im Befehl WseReturn() mit der Option WseNoParse verhindert werden. Soll zum Beispiel aufgrund eines Requests eine binäre Datei oder eine Datei, die keine C16.-Befehle enthält, an den Browser gesendet werden, kann mit der Option WseNoParse die Datei schneller übertragen werden, da ein Durchsuchen nach C16.-Befehlen entfällt.



Die Kommandos müssen in Großbuchstaben angegeben werden, damit sie vom Client erkannt werden.

Folgende Befehle werden vom Client ausgewertet:

C16.URL URL zurückgeben

C16.CALL Prozedur aufrufen

C16.UID ID des Web-Benutzers zurückgeben

**C16.CALL(alpha1[, alpha2]) :**

**alpha**

Prozedur oder Funktion aufrufen

Prozedur- und

**alpha1** Funktionsname

**alpha2** Parameter der Prozedur  
oder Funktion

**Resultat** alpha Rückgabewert der  
Funktion

**Siehe** Verwandte Befehle

Mit diesem Kommando wird eine Prozedur in der CONZEPT 16-Datenbank aufgerufen. Die Funktion wird in (alpha1) mit <Prozedurname>:<Funktionsname> angegeben. Optional können anschließend durch Kommas getrennt Argumente übergeben werden.

Wird nur der Prozedurname angegeben, wird die Funktion main der Prozedur aufgerufen.

Die Argumente werden nicht im Funktionskopf deklariert, sondern werden über den Befehl WseArg() in der Funktion ermittelt. Der Rückgabewert wird in der Funktion mit dem Befehl WseReturn() festgelegt. Der Befehl kann in einem Dokument mehrfach hintereinander aufgerufen werden. Die so zurückgegebenen Zeichenketten werden an der Stelle des C16.CALL-Aufrufs in die Seite eingefügt. Enthält die Funktion kein WseReturn(), wird der Aufruf durch eine leere Zeichenkette ersetzt.

### Beispiel

```
C16.CALL (WebText:Banner, 'DefaultPage', 1)
```



Innerhalb von Prozeduren, die durch die Web-Schnittstelle aufgerufen werden, dürfen nur Befehle, die mit dem Symbol versehen sind, verwendet werden. Die speziellen Befehle der Web-Schnittstelle werden im Abschnitt Befehle der Web-Schnittstelle erläutert.

**C16.UID() : alpha**

**ID des Web-Benutzers einfügen**  
ID des  
Resultat alpha

Web-Benutzers

Siehe [\*\*Verwandte Befehle\*\*](#)

Dieses Kommando gibt die Benutzer-ID des Web-Benutzers zurück. Die Benutzer-ID wird bei jedem Link (sofern er mittels ([C16.URL\(\)](#) erzeugt wird) automatisch mit angegeben, um die URL, die an den Client gesendet wird, einem Benutzer zuordnen zu können. C16.UID() wird daher nur in selbst definierten Links und beim Versenden von Formularen benötigt, wenn in der Konfigurationsdatei der Web-Schnittstelle der Eintrag web\_uid\_mode auf query gesetzt ist.

Bei Formularen sind die Parameter method und action des form-Tags entscheidend. Standardmäßig werden die Inhalte der Formularfelder an die Ursprungs-URL der HTML-Seite gesendet. In dieser URL ist aber die Benutzer-ID des Web-Benutzers nicht enthalten. Die Daten können also keinem Web-Benutzer zugeordnet werden. In einem solchen Fall kann ein verstecktes Eingabefeld in das Formular integriert werden, dass die benötigten Informationen enthält. Das Feld muss den Namen C16UID und den Rückgabewert von C16.UID() als value besitzen. Der Rückgabewert von C16.UID() dient zur Identifizierung des Benutzers.

Als action kann eine URL angegeben werden, an welche die Formulardaten gesendet werden. Dies funktioniert aber nur, wenn als Methode post angegeben wurde. Wird die Methode get verwendet, wird der Inhalt der Formularfelder als Parameter der URL angehängt. Dies hat zur Folge, dass die Benutzer-ID durch die Feldinhalte des Formulars überschrieben wird.

### Beispiel

```
<form><input type="hidden" name="C16UID" value="C16.UID()">...</form>
```

oder

```
<form method="post" action="C16.URL(abs)"> ...</form>
```



Die speziellen Befehle der Web-Schnittstelle werden im Abschnitt [Befehle der Web-Schnittstelle](#) erläutert.

## Kontakt

**C16.URL(alpha1[, alpha2]):**

**alpha**

URL der Web-Applikation einfügen

Bestandteile der URL

absolute URL bestehend  
aus Hostname,  
Skriptpfad und

Benutzer-ID

<b>alpha1</b>	<b>host</b>	Hostname
	<b>relative</b>	Skriptpfad und
		Benutzer-ID
	<b>static</b>	Hostname und der Skriptpfad

<b>alpha2</b>	Argumente im Query-String
	<b>alpha</b> URL innerhalb der Applikation
<b>Resultat</b>	<u>          </u>

Siehe [Verwandte Befehle](#)

Mit diesem Kommando wird die URL der CONZEPT 16-Web-Applikation in die HTML-Seite eingefügt. Der Befehl wird verwendet, um unabhängig von der Installation Pfade zu einzelnen Dokumenten beziehungsweise Web-Seiten innerhalb der Applikation angeben zu können. Der Parameter (alpha1) bestimmt dabei, welche Bestandteile eingefügt werden:

**absolute** - Die vollständige URL bestehend aus Hostname, Skriptpfad und Benutzer-ID wird zurückgegeben:

**http://www.vectorsoft.de/scripts/c16\_web.dll/C16UID.J1VE2OGBVUSOD47U2DFW67IC**

**host** - Der Hostname wird zurückgegeben:

**http://www.vectorsoft.de**

**relative** - Der Skriptpfad und die Benutzer-ID wird zurückgegeben:

**/scripts/c16\_web.dll/C16UID.J1VE2OGBVUSOD47U2DFW67IC**

**static** - Der Hostname und der Skriptpfad werden zurückgegeben:

**http://www.vectorsoft.de/scripts/c16\_web.dll**

Im Parameter (alpha2) können Parameter angegeben werden, die am Ende des Pfads angehängt werden. Parameter werden in der Notation Parametername=Wert angegeben. Mehrere Parameter werden durch Kommas getrennt. In der URL werden diese Parameter dann folgendermaßen angehängt:

**?Parameter1=Value1%Parameter2=Value2...**

## Kontakt

Befehle der Web-Schnittstelle

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 entsprechende Befehle enthalten.

## 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

## Kontakt

[WseRetString](#)  
[WseRetText](#)  
[WseSendAfter](#)  
[WseTerm](#)  
[WseUserHTML](#)  
[WseUserInit](#)  
[WseUserProc](#)  
[WseUserTerm](#)

## Kontakt

**Web-Schnittstelle - Erstellung von HTML-Seiten**

**Hindernisse beim Erstellen von HTML-Seiten**

Befehle der  
Web-Schnittstelle,

Siehe Fehlermeldungen  
der  
Web-Schnittstelle

HTML-Editoren pflegen vermeintlich fehlerhaften HTML-Code zu "berichtigen", was unter Umständen Konflikte mit C16.-Anweisungen verursachen kann. Im einfachsten Fall wird aus C16.CALL(Line) <p>C16.CALL(Line)</p> (dies ist allerdings ein harmloser Fall). Innerhalb von Tabellen kann es zu Veränderungen kommen, die von einem Browser nicht mehr korrekt interpretiert werden können.

```
<table border="0">  C16.CALL(Line,1)  C16.CALL(Line,2)  C16.CALL(Line,3)  C16.CALL(Line,4)</table>
```

**Der obige Text wird unter Einwirkung von FrontPage in folgendes Gebilde umgewandelt:**

```
<table border="0">  <tr>    <td>C16.CALL(Line,1)</td>  </tr>  <tr>    <td>C16.CALL(Line,2)</td>
```

**Um diesen Effekt zu vermeiden setzt man das (Standard) HTML-Element < ein (für die Unterbringung von Scripts gedacht). Statements, die in diese Klammer eingeschlossen sind, werden von Editoren nicht bewertet.**

**Die öffnende Klammer <% muss unmittelbar vor dem C16-Schlüsselwort stehen; die schließende %> unmittelbar hinter der schließenden Klammer:**

```
<%C16.CALL(Zeile,1)%>
```

**Die Statements <% und %> werden vom Client zusammen mit dem Funktionsaufruf aus dem Text entfernt.**

## **Kontakt**

**Web-Schnittstelle - Fehlermeldungen des Clients**

**Fehlermeldungen der Web-Schnittstelle**

**Fehlermeldungen werden auf zwei unterschiedlichen Wegen übermittelt. Zum einen muss der Anwender des Browsers über den Fehler informiert werden. Zu diesem Zweck wird eine Fehlerseiten an den Browser übermittelt. Zum anderen muss eine detaillierte Fehlermeldung in einer Log-Datei eingetragen werden, um eine Fehleranalyse zu ermöglichen.**

**Erstellung von HTML-Seiten**

**Web-Anbindung einer Datenbank**

## Kontakt

### Web-Schnittstelle - Rückgabe von Fehlerseiten

#### Rückgabe einer Fehlerseite an den Browser

Kommt es zu einem Laufzeitfehler, kann an den Browser eine Fehlerseite übermittelt werden. Die Fehlerseiten können für jede Applikation in einem eigenen Verzeichnis abgelegt werden. Das Verzeichnis wird in der Konfigurationsdatei im Eintrag web\_error\_path festgelegt. Ist kein solcher Eintrag vorhanden oder konnte die Applikation nicht identifiziert werden, wird eine Fehlerseite aus dem Verzeichnis der Datei c16\_web.dll gelesen.

Ist ein Fehler aufgetreten, wird zunächst nach einer CONZEPT 16 spezifischen

Fehlerseite gesucht. Der Name der Seite setzt sich dabei wie folgt zusammen:

error\_c16\_<nr>.htm. Wobei <nr> der Betrag der entsprechenden Fehlernummer ist.

Soll zum Beispiel bei einer Division durch Null (Fehlerwert -181) eine bestimmte Seite zurückgegeben werden, muss in dem Fehlerverzeichnis eine Seite mit dem Namen error\_c16\_181.htm erstellt werden. Folgende Laufzeitfehler werden von dem System generiert:

#### Code Beschreibung

- 1 ErrGeneric
- 20 ErrFsiNoFile
- 21 ErrFsiNoPath
- 22 ErrFsiOpenOverflow
- 23 ErrFsiAccessDenied
- 24 ErrFsiHdlInvalid
- 25 ErrFsiDriveInvalid
- 26 ErrFsiCurrentDir
- 27 ErrFsiSharingViolation
- 28 ErrFsiLockViolation
- 29 ErrFsiOpenFailed
- 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

## Kontakt

-186 ErrNoKey

-187 ErrNoLink

-188 ErrValueInvalid

-189 ErrNoKeyFld

-190 ErrNoLinkFld

-191 ErrHdlInvalid

-192 ErrNoArgument

-193 ErrLinkInvalid

-194 ErrFileInvalid

-50025 Die maximale Anzahl der Verbindungen (web\_max\_connections) ist überschritten.

-50026 Das eingetragene Sessionlimit (Web\_max\_sessions) wurde überschritten.

-50032 Das eingetragene Verbindungslimit der Applikation (c16\_max\_connections) wurde überschritten.

-50034 Die Applikation konnte nicht identifiziert werden.

-50051 Die verwendete Request-Methode wird nicht unterstützt. Es sind nur die Methoden GET, POST und HEAD zulässig.

-50053 Die Daten des Request konnten nicht gelesen werden.

-50054 Der Benutzer wurde aufgrund eines Timeouts entfernt.

Wird zum entsprechenden Fehlerwert keine Seite gefunden, wird aufgrund des HTTP-Statuscodes im gleichen Verzeichnis eine Datei mit dem Namen `error_http_<nr>.htm` gesucht. <nr> entspricht dabei dem ermittelten HTTP-Status. Der zurückgegebene Status hängt wesentlich von der verwendeten Protokollversion ab. Welche Statuscodes zurückgegeben werden, können dem [RFC 1945](#) und [RFC 2068](#) entnommen werden. Vom Client selbst werden nur folgende Statuscodes zurückgegeben:

Es wurde ein Request mit einer anderen Methode als GET, POST oder HEAD gestellt (nur bei HTTP/1.0)

Die angegebene Applikation wurde nicht gefunden

Es wurde ein Request mit einer anderen Methode als GET, POST oder HEAD gestellt (nur bei HTTP/1.1)

Dieser Status wird zurückgegeben, wenn eine Prozedur einen Laufzeitfehler generiert oder nicht genügend Speicher allokiert werden konnte

Dieser Status wird zurückgegeben, wenn sich zu viele Benutzer angemeldet haben

**Beispiel:**

`error_http_500.htm`

Da keine weiteren Statuscodes von der Schnittstelle zurückgegeben werden, müssen auch nur für diese entsprechende Fehlerseiten erstellt werden. Wird auch diese Datei nicht gefunden, wird die Datei `error.htm` zurückgegeben. Diese Datei kann verwendet werden, um unbekannte oder unspezifische Fehlermeldungen an den Browser zu übergeben. Existiert auch diese Datei nicht, generiert der Client eine Fehlerseite,

## **Kontakt**

**welche den HTTP-Status und den Laufzeitfehlercode enthält:**

## Kontakt

### Web-Schnittstelle - Log-Datei

#### Einträge in die Log-Datei

Die Fehlermeldungen des Clients werden in entsprechende Log-Dateien eingetragen. Für jeden Tag wird dabei eine eigene Log-Datei angelegt. Das Verzeichnis für die allgemeine Log-Datei wird in der Datei **c16\_web.ini** in dem Eintrag **logfile** angegeben. Existiert kein solcher Eintrag, wird die Datei im gleichen Verzeichnis, wie die **c16\_web.dll** angelegt. Der Name der Datei setzt sich aus **c16\_web** oder dem Namensteil im Eintrag **logfile**, dem Datum (im Format **JJJJMMTT**) und der Erweiterung **.log** zusammen.

Für jede Applikation existiert zusätzlich eine eigene Log-Datei, deren Verzeichnis in der Konfigurationsdatei des Clients definiert werden kann. Die Eintragungen in die Dateien gliedern sich in folgende Bereiche:

#### Information (i)

#### Fehler beim Starten (s)

#### Fehler im laufenden Betrieb (e)

#### Information (i)

Service start requested	Der Client soll gestartet werden.
Service started	Der Client ist gestartet worden.
Service stop requested	Der Client soll entladen werden.
Service stopped	Der Client wurde entladen.
Session begin	Ein Web-Benutzer hat sich angemeldet. Hinter dem Eintrag wird die Web-Benutzer-ID und die IP-Nummer des Benutzers angegeben.

#### Session end

#### Fehler beim Starten (s)

Service start failed	Beim Starten des Clients ist ein Fehler aufgetreten.
Out of memory	Der Client konnte nicht den benötigten Speicher allozieren.
Missing configuration file	Die Schnittstelle konnte die Konfigurationsdatei nicht finden. Der Name der Datei wird in Klammern hinter der Meldung angegeben.
Missing default section in configuration file	Beim Lesen der Konfigurationsdatei wurden keine oder nicht alle notwendigen Einträge gefunden.
User Manager thread start failed	Der Thread für den Benutzer-Manager konnte nicht gestartet werden.
Application Manager thread start failed	Der Thread für den Applikations-manager konnte nicht gestartet werden.
Application Manager out of memory	Der Applikationsmanager konnte nicht den benötigten Speicher allozieren. Die benötigte Speichermenge wird in Klammern angegeben.

## Kontakt

<b>Fehler im laufenden Betrieb (e)</b>	
<b>Session refused - connection limit reached</b>	<b>Das in der Konfigurationsdatei eingetragene Benutzerlimit des Clients (<u>web_max_connections</u>) wurde überschritten. Das aktuelle Limit wird in Klammern angegeben.</b>
<b>Session refused - connection limit from &lt;IP-Adresse&gt; reached</b>	<b>Das in der Konfigurationsdatei angegebene Verbindungslimit (<u>web_max_sessions</u>) pro Quell-IP-Adresse wurde überschritten. Das aktuelle Limit wird in Klammern angegeben.</b>
<b>Session refused - database user limit reached</b>	<b>Das Benutzerlimit der Datenbank wurde überschritten. Das Benutzerlimit wird in Klammern angegeben.</b>
<b>Method not allowed</b>	<b>An den Client ist ein Request mit einer Methode gesendet worden, die nicht unterstützt wird. Es werden nur die Methoden GET, HEAD und POST unterstützt. Die Methode wird in Klammern hinter dem Eintrag ausgegeben.</b>
<b>Out of memory &lt;POST data&gt;</b>	<b>Es konnte nicht genügend Speicher allokiert werden, um alle Daten des Requests zu speichern. Die benötigte Speichermenge wird in Klammern hinter dem Eintrag angegeben.</b>
<b>Client POST data read failed</b>	<b>Bei der Methode POST konnten die Daten nicht gelesen werden.</b>
<b>Timeout on client request</b>	<b>Ein Web-Benutzer stellte einen Request, während für den gleichen Web-Benutzer noch ein Request bearbeitet wurde. Diese Meldung kann umgangen werden, wenn länger dauernde Requests mit der Option <u>WseSendAfter</u> beantwortet werden.</b>
<b>Procedure runtime error</b>	<b>Beim durchführen einer Prozedur trat ein Laufzeitfehler auf. Die Nummer des Fehlers und der Name der Prozedur werden nach dem Eintrag angegeben.</b>
<b>Database connect failed</b>	<b>Es konnte keine Verbindung zur Datenbank aufgenommen werden. In Klammern steht der entsprechende Fehlerwert.</b>

## Kontakt

### SOA-Service

#### Beschreibung des SOA-Service

Siehe [Blog](#)



Eine Auflistung der Betriebssysteme, auf denen der SOA-Service betrieben werden kann, finden Sie in den [Systemvoraussetzungen](#).

Der SOA-Service ist eine Anwendungsumgebung, in der unter einem einzigen Systemdienst mehrere voneinander unabhängige Tasks laufen. Jeder Task läuft in einem eigenen Betriebssystemprozess und kann separat gestartet oder gestoppt werden. Der Prozess des Systemdienstes selbst ist dabei nur für das Starten und Stoppen der definierten Tasks zuständig. Ein Task kann entweder zeitgesteuert (Betriebsart TIME) oder als Socketserver (Betriebsart SOCKET) betrieben werden. Die Task und die Betriebsart des Task werden bei der [Konfiguration des SOA-Service](#) festgelegt.

Der SOA-Service benötigt auf dem Rechner, auf dem er läuft, einen Benutzer, sofern nicht bereits ein anderer Client gestartet ist.

#### Betriebsart TIME

In dieser Betriebsart wird die Ereignisprozedur des Task in zeitlichen Abständen wiederholt aufgerufen. Der Zeitpunkt des nächsten Aufrufs kann innerhalb der Ereignisprozedur verändert werden. Während der Ausführung der Ereignisprozedur finden keine weiteren Ereignisse statt, es ist somit maximal eine Verarbeitungsinstanz aktiv. Benötigt die Verarbeitung einen längeren Zeitraum, sollte ein oder mehrere Jobs (siehe unten) gestartet werden.

#### Betriebsart SOCKET

In dieser Betriebsart wartet der Task passiv auf eingehende Verbindungen auf einem bestimmten TCP/IP-Port. Beim Zustandekommen einer Verbindung wird die Ereignisprozedur des Tasks gestartet, die dann die Kommunikation mit der Gegenstelle durchführt. Nach dem Ende der Prozedur wird die Verbindung in der Regel automatisch geschlossen. Sie kann aber auch für einen bestimmte Zeit (Keep-Alive-Time) noch bestehen bleiben, um beispielsweise weitere Anfragen der Gegenstelle zu verarbeiten. Wenn während der Ausführung der Ereignisprozedur weitere Verbindungen entstehen, werden weitere Instanzen der Ereignisprozedur gestartet, es können somit mehrere Ereignisprozeduren parallel ablaufen.

Ein socketbasierter SOA-Service kann optional mit Verschlüsselung unter Verwendung der Transport Layer Security betrieben werden. Dafür ist ein entsprechendes SSL/TLS-Zertifikat notwendig. Das Zertifikat muss im PEM-Format vorliegen. Die Datei für das Zertifikat wird in den Einstellungen des Tasks angegeben (siehe [SOA-Service - Konfigurationsdatei](#)). Die Daten werden beim Start des Tasks verwendet, treten dabei Fehler auf, befinden sich entsprechende Eintragungen in der [Protokolldatei des Tasks](#). Nach dem Starten des Tasks werden nur noch Verbindungen akzeptiert, die SSL 3.0 oder TLS 1.x entsprechen.

#### Jobs

Innerhalb einer Ereignisprozedur können sogenannte Jobs gestartet werden. Ein Job

## Kontakt

führt eine Prozedur aus, die parallel zur Ereignisprozedur durchgeführt wird und auch nach dem Ende der Ereignisprozedur noch weiterlaufen kann. Ein Job kann innerhalb des Task-Prozesses als separater Thread oder alternativ als eigenständiger Betriebssystemprozess durchgeführt werden. Die Ereignisprozedur kann mit der Job-Prozedur einen Datenaustausch mit Hilfe von MSX-Befehlen realisieren. Eine Job-Prozedur benötigt einen eigenen Datenbankbenutzer.

## Session-Management

Bei socketbasierten Tasks entsteht häufig die Notwendigkeit, über mehrere Anfragen hinweg einen Verarbeitungskontext zu haben, der nicht implizit durch das verwendete Protokoll gegeben ist. Bei SMTP gibt es beispielsweise für die gesamte Session nur eine Verbindung die auch nur durch einen einzigen Prozeduraufruf verarbeitet wird. Bei HTTP dagegen handelt es sich um ein "zustandsloses" Protokoll, bei dem Sessions außerhalb des Protokolls verwaltet werden müssen.

Innerhalb eines SOA-Tasks lässt sich ein Verarbeitungskontext erzeugen, in dem Deskriptoren und globale Datenbereiche nach einem Prozeduraufruf gesichert und beim nächsten Prozeduraufruf wiederhergestellt werden. Ein Verarbeitungskontext wird nach einem konfigurierbaren Timeout (siehe session timeout max) automatisch durch den Task gelöscht. Alle weiteren Einstellungen zum Session-Management sind im Abschnitt Konfigurationsdatei des SOA-Service beschrieben.

Für die Handhabung von Sessiondaten steht der Befehl SvcSessionControl() zur Verfügung.

## Betriebsart DRIVE

Mit dieser Betriebsart wird der Laufwerkstreiber gestartet. Mit dem Laufwerkstreiber wird ein Laufwerk definiert. Alle Aktionen auf dem Laufwerk lösen eine Nachricht an eine definierte Applikation aus. Die Applikation muss die angeforderten Informationen zur Verfügung stellen.

## Kontakt

**Installation des SOA-Services unter Windows**

**Beschreibung der Installation des SOA-Services unter Windows**

**Die Dateien der SOA-Service werden über die CONZEPT 16 Installationsroutine eingerichtet.**

**Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \SOA.**

*Dateien des SOA-Service*

**c16\_soa\_cmd\_win.exe** Script-Utility

**c16\_soa\_drv\_win.dll** Laufwerkstreiber

**c16\_soa\_svc\_win.dll** System-Dienst

**c16\_soa\_tsk\_w32.dll** Taskprogramm (32-Bit)

**c16\_soa\_tsk\_w64.dll** Taskprogramm (64-Bit)

**c16\_soa\_tsk\_win.dll** Taskprogramm (32-Bit mit grafischer Erweiterung)

---

**Zusätzlich benötigte Dateien in diesem Verzeichnis:**

**c16.tla** Token-Übersetzungstabelle

**c16.vra** Ressourcendatei

**c16\_bar2d\_w32.dll** Barcode-Bibliothek (32-Bit)

**c16\_chart\_w32.dll** Chart-Bibliothek (32-Bit)

**c16\_coded\_w32.dll** CodeEdit-Bibliothek (32-Bit)

**c16\_diff\_w32.dll** Difference-Bibliothek (32-Bit)

**c16\_graph\_w32.dll** Graphics-Bibliothek (32-Bit)

**c16\_icu\_w32.dll** Unicode-Bibliothek (32-Bit)

**c16\_obj\_w32.dll** GUI-Objekte (32-Bit)

**c16\_pdf\_w32.dll** PDF-Bibliothek (32-Bit)

**c16\_pdfx\_w32.dll** PDF-Anzeige-Bibliothek (32-Bit)

**c16\_res\_w32.dll** GUI-Ressourcen (32-Bit)

**c16\_ssl\_w32.dll** SSL-Bibliothek (32-Bit)

**c16\_ssl\_w64.dll** SSL-Bibliothek (64-Bit)

**c16\_sys\_w32.dll** System-Objekte (32-Bit)

**c16\_xml\_w32.dll** XML-Bibliothek (32-Bit)

**c16\_xml\_w64.dll** XML-Bibliothek (64-Bit)

**c16\_zip\_w32.dll** ZIP-Bibliothek (32-Bit)

**c16\_zip\_w64.dll** ZIP-Bibliothek (64-Bit)

**Die Konfiguration des SOA-Service wird über Konfigurationsdateien vorgenommen. Nach der Installation ist im Datenverzeichnis von CONZEPT das Unterverzeichnis SOA eingerichtet. In diesem Verzeichnis muss die Konfigurationsdatei des Dienstes als auch die Konfigurationsdateien der einzelnen Tasks stehen.**

**Nach der Installation kann der SOA-Service Systemdienst als auch die einzelnen Tasks über das Control-Center gestartet und gestoppt werden. Das Control-Center kann auch einzelne Tasks anlegen und entfernen (siehe SOA-Service - Konfigurationsdatei). Alternativ können Tasks über das Script-Utility abgefragt, gestartet und gestoppt werden.**

## Kontakt

Je nach Betriebssystem und Konfiguration wird eine der c16\_soa\_tsk\_w???.dll-Dateien verwendet:

**c16\_soa\_tsk\_w32.dll - 32-Bit Betriebssystem oder 64bit = N und**

**c16\_proc\_extended = N**

**c16\_soa\_tsk\_w64.dll - 64-Bit Betriebssystem und 64bit = Y**

**c16\_soa\_tsk\_win.dll - 32-Bit Betriebssystem oder 64bit = N und**

**c16\_proc\_extended = Y oder nicht gesetzt**

Kommt es zum Absturz des SOA-Services, wird bei den Dateien c16\_soa\_tsk\_w32.dll und c16\_soa\_tsk\_w64.dll eine Minidump-Datei (c16\_soa\_tsk\_w??\_\*.mdmp) erzeugt. Diese befindet sich in einem der Verzeichnisse in der folgenden Reihenfolge:

Pfad, der durch die TMP-Umgebungsvariable definiert ist.

Pfad, der durch die TEMP-Umgebungsvariable definiert ist.

Pfad, der durch die USERPROFILE-Umgebungsvariable definiert ist.

Windows-Verzeichnis.

## Kontakt

### Installation des SOA-Services unter Linux

#### Beschreibung der Installation des CONZEPT 16-SOA-Services unter Linux

Die Programmdateien des SOA-Services unter Linux werden über die CONZEPT 16 Installationsroutine erstellt. Die Programmdateien und die Installationsroutine für den befinden sich anschließend im Installationsverzeichnis unter \SOA\Linux im komprimierten TAR-Archiv **c16\_soa\_lnx.tar.gz**. Dieses wird auf ein Linux-System transferiert und dort mit dem folgenden Kommando dekomprimiert und entpackt:

```
tar -zxvf c16_soa_lnx.tar
```

Dies geht auch in einzelnen Schritten mit den Kommandos:

```
gzip -d c16_soa_lnx.tar.gztar -xvf c16_soa_lnx.tar
```

#### *Inhalt des Archivs*

**c16\_soa\_cmd\_l32** Script-Utility für 32-Bit

**c16\_soa\_cmd\_l64** Script-Utility für 64-Bit

**c16\_soa\_svc\_l32** Serviceprozess (Daemon) 32-Bit

**c16\_soa\_svc\_l64** Serviceprozess (Daemon) 64-Bit

**c16\_soa\_tsk\_l32** Task für 32-Bit

**c16\_soa\_tsk\_l64** Task für 64-Bit

**c16.vra** Ressourcendatei

**c16\_error** Liste der behobenen Fehler

#### Starten des SOA-Services

Nach dem entpacken kann der SOA-Service gestartet werden (siehe [Starten und Stoppen des SOA-Services unter Linux](#)).



Der SOA-Service kann so eingerichtet werden, dass er beim Start des Betriebssystems automatisch mitgestartet wird. Details zum Boot-Konzept der verschiedenen Distributionen befinden sich in der entsprechenden Dokumentation der Distribution.

#### Update des SOA-Services

Zum Update des SOA-Services wird dieser zunächst beendet. Dann können die neuen Programmdateien in das Verzeichnis des SOA-Services entpackt und in der neuen Version gestartet werden.

## Kontakt

**SOA-Service - Starten und Stoppen unter Linux** Starten und Stoppen des SOA-Services unter Linux Starten des SOA-Services

Der **SOA-Service** wird auf allen 32- Bit Linux-Plattformen mit dem Programm **c16\_soa\_svc\_l32** und auf allen 64-Bit Linux-Plattformen mit dem Programm **c16\_soa\_svc\_l64** gestartet. Befindet sich der aktuelle Pfad nicht in der Path-Variable des Betriebssystems, muss ./ der Anweisung vorangestellt werden. Nach dem Starten des Service-Prozesses wird auf der Konsole folgendes ausgegeben:

```
CONZEPT 16 SOA Service Client Version 5.8.12 - Linux (64-bit) (C) Copyright by vectorsoft AG
```

Der Service-Prozess wird unter Linux als Daemon gestartet, das heißt der Service-Prozess hängt sich direkt an den Init-Prozess des Betriebssystems an und ist somit von den Benutzer-Sessions unabhängig. Der Programmaufruf von **c16\_soa\_svc\_l32** beziehungsweise **c16\_soa\_svc\_l64** kehrt sofort zurück. Beim Starten kann aber zusätzlich die Option **-wait** angegeben werden. In diesem Fall kehrt der Aufruf erst zurück, wenn der Service-Prozess erfolgreich gestartet wurde, jedoch maximal nach 30 Sekunden.

**Stoppen des SOA-Services**

Der SOA-Service wird mit der Anweisung "**c16\_soa\_svc\_l32 -down**" beziehungsweise "**c16\_soa\_svc\_l64 -down**" angehalten. Beim Stoppen des SOA-Services kann ebenfalls das Argument **-wait** angegeben werden. Die Anweisung wartet dann maximal 20 Sekunden, bis der SOA-Service beendet ist.



Das Argument **-wait** sollte in Skripts in jedem Fall verwendet werden, um eine unbeaufsichtigte Durchführung zu ermöglichen.

## Kontakt

### Konfiguration des SOA-Service

#### Einrichten des SOA-Service und der Tasks

Siehe

**Beispiel -**  
c16\_soa.cfg.

**Beispiel -**

In diesem Abschnitt wird sowohl die Konfiguration des SOA-Service, als auch die Konfiguration der Tasks des Service beschrieben. Die Konfiguration des Service wird in der Datei **c16\_soa.cfg** im **Datenverzeichnis** gespeichert. Hier werden alle Tasks eingetragen. Im gleichen Verzeichnis werden auch die Konfigurationsdateien der Tasks abgelegt. Die Dateinamen bestehen aus dem Namen des Tasks mit der Dateierweiterung **.cfg**.



Bei der Angabe von Zeiten in einem Eintrag, müssen ganze Zahlen mit einer Zeiteinheit angegeben werden. Als Einheiten stehen Tage (d), Stunden (h), Minuten (m), Sekunden (s) und Millisekunden (ms) zur Verfügung. Dabei muss jeweils die höchste Einheit angegeben werden, die den Zeitraum abdeckt (z. B. 1m statt 60s). Die Einheiten können auch miteinander kombiniert werden, zum Beispiel 1h30m.

### Konfiguration des Dienstes

In der Datei **c16\_soa.cfg** können mehrere Anwendungstasks definiert werden. Jeder Task wird mit seinem Namen in eckigen Klammern eingeleitet. Der Name kann maximal 128 Zeichen lang sein. Danach folgen die Konfigurationsparameter des Tasks. Die Konfiguration der Task kann auch im **Control-Center** durchgeführt werden. Für den Systemdienst selbst existieren keine einstellbaren Parameter.

#### **;(Semikolon) Kommentar**

<u>mode</u>	Betriebsart des Tasks
<u>autostart</u>	Automatischer Start
<u>autorevive</u>	Automatischer Neustart bei Absturz
<u>64bit</u>	64-bit Task
<u>start_delay</u>	Start-Verzögerung
<u>description</u>	Beschreibung

#### **;(Semikolon) - Kommentar**

Mit dem Zeichen ";" wird eine Kommentarzeile eingeleitet. Die gesamte restliche Zeile wird nicht mehr ausgewertet.

#### **mode - Betriebsart**

Über diesen Parameter wird die Betriebsart des Tasks bestimmt. Es sind die Betriebsarten **TIME**, **SOCKET** und **DRIVE** verfügbar.

#### **autostart - Automatischer Start**

Mit diesem optionalen Parameter kann bestimmt werden, ob nach dem Start des Dienstes der Task ebenfalls automatisch gestartet wird (**autostart = Y**) oder nicht (**autostart = N**).

#### **autorevive - Automatischer Neustart bei Absturz**

## Kontakt

Mit diesem optionalen Parameter kann bestimmt werden, ob ein Task automatisch neu gestartet werden soll, wenn er abgestürzt ist (autorevive = Y) oder nicht (autorevive = N). Das Startintervall beträgt 10 Sekunden. Der Standardwert ist N.

### 64bit - 64-bit Task starten

Mit diesem optionalen Parameter kann bestimmt werden, ob der 32-bit (64bit = N) oder 64-bit (64bit = Y) Task gestartet werden soll.

### start\_delay - Startverzögerung

Falls der Parameter autostart = Y definiert ist, kann mit diesem optionalen Parameter die Zeitverzögerung festgelegt werden, die zwischen Start des Systemdienstes und dem Start des Tasks stattfinden soll. Die Zeitspanne kann im Bereich von 0 bis 5 Minuten liegen. Bei einem Wert von 0 versucht der Task unter Umständen sofort eine Verbindung mit dem CONZEPT 16-Server aufzunehmen. Da die Reihenfolge, in der die Dienste gestartet werden, nicht garantiert werden kann, kann es zu einem Fehler kommen, wenn der lokale CONZEPT 16-Server noch nicht gestartet ist. Durch den Eintrag einer Verzögerung wartet der Systemdienst die hier angegebene Zeit, um dem Betriebssystem Zeit zu geben, den Server zu starten.

### description - Beschreibung

Durch diesen optionalen Parameter kann ein Beschreibungstext für den Task abgelegt werden. Die Beschreibung kann bis zu 500 Zeichen lang sein.

Beispiel:

```
[Abfragen]mode      = SOCKETautostart    = Yautorevive   = Ystart_delay = 5sdescription = 'Abfrag'
```

## Konfiguration eines Tasks

Im selben Verzeichnis wie c16\_soa.cfg befindet sich pro Task eine eigene Konfigurationsdatei mit dem Namen <task>.cfg. In dieser Datei können die folgenden Parameter definiert werden:

c16 server

c16 database

c16 user

c16 password

c16 procedure

c16 proc cache kb

c16 proc library

c16 connection max

c16 connection shared

c16 connection timeout

c16 proc extended

request\_delay\_min

request\_delay\_max

## Kontakt

session\_max  
session\_timeout\_max  
session\_timeout\_min  
alert\_mail\_server  
alert\_mail\_port  
alert\_mail\_user  
alert\_mail\_password  
alert\_mail\_from  
alert\_mail\_to  
alert\_mail\_to\_sms  
alert\_mail\_proxy  
alert\_mail\_tls  
alert\_lines  
alert\_interval  
alert\_test  
applogoptions  
applogpath  
time\_delay\_default  
time\_delay\_error  
socket\_port  
socket\_ip  
socket\_timeout  
socket\_keepalive  
ip\_connection\_limit  
tls\_certificate\_file  
tls\_private\_key\_file  
tls\_private\_key\_password  
tls\_security\_level

Alle Parameter, die in der Datei angegeben werden müssen, sind mit einem \* gekennzeichnet. Die Aufstellung der Parameter unterteilt sich in folgende Abschnitte:

### Einstellungen der Datenbankverbindung

Session-Management

Automatischer Mailversand

Minidump-Generierung

Betriebsart TIME

Betriebsart SOCKET

Verschlüsselte Kommunikation über SOCKET-basierte Tasks

Betriebsart DRIVE

### Einstellungen der Datenbankverbindung

**c16\_server \***

## Kontakt

In diesem Eintrag wird der CONZEPT 16-Server eingetragen. Der Eintrag erfolgt in Form der IP-Adresse oder des IP-Namens des Serverrechners. Zur Verbindung wird immer das Protokoll TCP/IP verwendet. Bei Verwendung der Hot-Standby-Option werden der Primär- und der Sekundärserver mit einem + getrennt angegeben.

```
c16_server = dbprimserv+dbsekser  
c16_database *
```

In diesem Eintrag wird die Datenbank eingetragen. Der Eintrag entspricht dem Namen der Datenbank, wie sie beim CONZEPT 16-Server eingetragen ist. Wurde kein symbolischer Name vergeben kann auch der Pfad und der Name der Datenbank angegeben werden.

```
c16_database = codelibrary  
c16_user und c16_password *
```

Die Anmeldung an die Datenbank erfolgt mit den hier angegebenen Benutzer und Kennwort. Hat der Benutzer kein Kennwort muss "" angegeben werden.

Enthält die Konfigurationsdatei eines SOA-Tasks ein unverschlüsseltes Kennwort das nicht leer ist, wird die Konfigurationsdatei automatisch mit dem verschlüsseltem Kennwort aktualisiert. Kennwörter sind dadurch nicht im Klartext enthalten.

```
c16_user = servicec16_password =
```



Das Passwort darf nicht von Anführungszeichen ("") eingeschlossen werden.

```
c16_procedure *
```

Dies ist der Name der Ereignis-Funktion in der Datenbank, die durch die Applikation gestartet wird. Die Angabe erfolgt in der Form "Prozedurname:Subfunktion". Der Funktion werden der Deskriptor des Service-Objekts und der Ereignistyp übergeben. Soll die main-Funktion einer Prozedur aufgerufen werden, muss lediglich der Name der Prozedur angegeben werden.

```
c16_procedure = SvcMain:EvtConnect  
c16_proc_cache_kb
```

In diesem Eintrag kann für die Verarbeitung der Prozedur-Cache eingetragen werden. Die Angabe erfolgt in KB und muss im Bereich von 256 bis 65536 (64 MB) liegen.

```
c16_proc_cache_kb = 2048  
c16_proc_library
```

In diesem Eintrag kann der Name einer zu ladenden DLL bzw. Shared Library angegeben werden. Der Befehl DllLoad() bezieht sich auf die

## Kontakt

zuvor geladene Bibliothek, wenn eine leere Zeichenkette als Name für die Bibliothek übergeben wird.

Der SOA-Task loggt die Meldung **Procedure library load failed**, falls das Laden der Bibliothek nicht erfolgreich durchgeführt wurde.

```
c16_proc_library = MyLibrary.dll  
c16_connection_max
```

In dieser Eigenschaft wird die maximale Anzahl von Verbindungen zur Datenbank angegeben. Eine Verbindung wird nur solange genutzt, bis die Ereignisfunktion beendet ist. In Abhängigkeit von **c16\_connection\_shared** wird die Verbindung danach entweder getrennt oder sie steht danach für das nächste Ereignis zur Verfügung.

Der Eintrag kann zwischen 1 und 1000 gewählt werden. Der hier eingetragene Wert entspricht der Anzahl von Ereignissen, die gleichzeitig bearbeitet werden können. Der Standardwert ist 500.

```
c16_connection_max = 10  
c16_connection_shared *
```

Dieser Parameter bestimmt, ob eine Datenbankverbindung nach der Durchführung der Ereignisfunktion beendet wird (**c16\_connection\_shared**

N) oder ob sie für weitere Ereignisse weiter zur Verfügung steht (**c16\_connection\_shared = Y**). In der Testphase ist es durchaus sinnvoll das Benutzer-Sharing abzuschalten, da zusammen mit dem Beenden der Verbindung auch der Inhalt des Prozedurcaches geleert wird. Im produktiven Einsatz kann das Benutzer-Sharing sowohl die Anzahl der benötigten Verbindungen reduzieren (siehe **request\_delay\_min** und **request\_delay\_max**) und die Verarbeitungsgeschwindigkeit erhöhen.

```
c16_connection_shared = Y  
c16_connection_timeout (Dieser Wert ist nur bei  
c16_connection_shared = Y von Bedeutung)
```

Wird eine Datenbankverbindung länger als die hier eingestellte Zeit nicht benutzt, wird sie automatisch beendet, d. h. der Datenbankbenutzer wird abgemeldet. Es können Werte von 10 Sekunden bis zu einer Stunde eingetragen werden. Ist dieser Wert nicht gesetzt, erfolgt der Timeout nach 5 Minuten.

```
c16_connection_timeout = 5m  
c16_proc_extended
```

Über diesen Eintrag wird gesteuert, ob der Task die Verwendung von Oberflächen- bzw. Druck-Objekten unterstützt (**c16\_proc\_extended = Y**) oder nicht (**c16\_proc\_extended = N**).



Diese Option wird ignoriert, wenn der Eintrag **64bit** der **c16\_soa.cfg** gesetzt ist.

## Kontakt

Damit ein Dienstprogramm mit dem Benutzer interagieren kann, muss dies bei den Einstellungen des Dienstes erlaubt werden. Standardmäßig wird die Verwendung von Oberflächen- und Druck-Objekten unterstützt.

c16\_proc\_extended = Y

-  Ist dieser Eintrag auf N gesetzt, können alle Befehle ausgeführt werden, die in der Dokumentation mit dem Symbol ⓘ versehen sind. Ist der Eintrag auf Y gesetzt, können zusätzlich dazu die Befehle mit dem Symbol ⓘ verwendet werden.

request\_delay\_min (Dieser Wert ist nur bei c16\_connection\_shared = Y von Bedeutung)

Wenn bei Eintritt eines neuen Ereignisses alle bestehenden Datenbankverbindungen der Applikation bereits mit der Bearbeitung von Ereignissen beschäftigt sind, wird nach Ablauf der hier eingetragenen Zeit eine neue Datenbankverbindung aufgebaut. Wird in diesem Zeitraum eine Datenbankverbindung frei, wird diese zur Bearbeitung des Ereignisses verwendet. Die Zeitspanne kann im Bereich von 0 bis 3 Sekunden liegen. Bei 0 wird sofort eine neue Verbindung eingerichtet. Ist der Wert nicht gesetzt, erfolgt Aufbau einer weiteren Datenbankverbindung nach 50 Millisekunden.

request\_delay\_min = 50ms

request\_delay\_max (Dieser Wert ist nur bei c16\_connection\_shared = Y von Bedeutung)

Kann keine weitere Datenbankverbindung eingerichtet werden

(c16\_connection\_max ist erreicht), wird maximal die hier eingestellte Zeit gewartet. Wird eine Datenbankverbindung in diesem Zeitraum frei, wird diese für das Ereignis verwendet. Kann auch nach dieser Zeit keine Verbindung benutzt werden, wird die Socket-Verbindung getrennt und der Fehler protokolliert. Es können Werte zwischen 100 Millisekunden und 10 Sekunden eingetragen werden.

request\_delay\_max = 3s

## Session-Management

In diesen Eintragungen werden die Einstellungen für das Sessionmanagement vorgenommen. Die Eintragungen sind nur dann relevant, wenn mit einer oder mehreren Sessions (siehe SvcSessionControl()) gearbeitet wird.

session\_max

Dies ist die maximal zulässige Anzahl von Sessions für diesen Task. Der Vorgabewert ist 0 (keine Beschränkung), zulässige Werte liegen im Bereich von 0 bis 100000. Beim Überschreiten des Limits liefert SvcSessionControl( SvcSessionCreate) das Resultat ErrLimitExceeded.

session\_max = 0

session\_timeout\_max

## Kontakt

Nach dieser Zeitspanne wird eine nicht mehr verwendete Session durch den Task gelöscht. Der Vorgabewert ist 20 Minuten (20m), zulässige Werte liegen im Bereich von 5 Sekunden (5s) bis 4 Stunden (4h).

`session_timeout_max = 20m`

`session_timeout_min` (Dieser Wert ist nur bei `session_max` ungleich 0 von Bedeutung)

Dies ist die minimale Zeitspanne, die eine Session nach dem Prozedurende erhalten bleibt. Sie ist nur relevant, wenn `session_max` ungleich Null ist. In diesem Fall werden beim Erreichen des Limits zunächst Sessions gelöscht, die länger als `session_timeout_min` nicht mehr verwendet wurden.

Der Vorgabewert ist 5 Minuten (5m), zulässige Werte liegen im Bereich von 5 Sekunden (5s) bis 1 Stunde (1h).

`session_timeout_min = 5m`

## Automatischer Mailversand

Erfolgt aufgrund einer Fehlermeldung ein Eintrag in der Log-Datei des SOA-Service, kann automatisch eine E-Mail an eine bestimmte Adresse versendet werden. In der Mail wird die Log-Datei mit versendet. Es werden maximal 64 KB verschickt. Einträge der Klassen Initialisieren (s) und Information (i) führen nicht zum Versenden einer E-Mail.

Zusätzlich können eine oder mehrere Mails nur mit der letzten Zeile der Log-Datei im "Betreff" versendet werden. Diese Mails enthalten keinen Mail-Text und sind für eine Übermittlung als SMS (via E-Mail-Gateway) besonders geeignet.

Die notwendigen Angaben zum Versenden der E-Mail werden in den folgenden Einträgen vorgenommen:

`alert_mail_server`

In diesem Eintrag wird der Name oder die IP-Adresse des Mail-Servers eingetragen. Der Mail-Server muss über das Protokoll SMTP auf dem, mit `alert_mail_port` definierten Port erreichbar sein.

`alert_mail_server = 192.168.0.12`

`alert_mail_port`

In diesem Eintrag wird der Port des Mail-Servers eingetragen. Der Mail-Server muss über das Protokoll SMTP auf diesem Port erreichbar sein. Ist diese Einstellung nicht angegeben, wird Port 25 verwendet.

`alert_mail_port = 25`

`alert_mail_user` und `alert_mail_password`

Sofern der verwendete Mailserver eine Authentifizierung erfordert, kann

## Kontakt

in diesen Einträgen der Benutzername (meist in der Form name@domain.tld) und das Kennwort angegeben werden. Der Mailserver muss das SMTP-Kommando "AUTH" in Verbindung mit einer der Methoden "PLAIN", "LOGIN" oder "CRAM-MD5" unterstützen. Sofern der Mailserver mehrere Methoden unterstützt, wird bevorzugt CRAM-MD5 verwendet, gefolgt von LOGIN.

```
alert_mail_user = name@vectorsoft.de
alert_mail_password = 2dhwC1GHNtM7
alert_mail_from
```

In diesem Eintrag wird die Absender-Adresse eingetragen. Die Mailadresse muss vom Mailserver akzeptiert werden. Aufgrund des Absenders muss es möglich sein, den Task, der den Fehler protokolliert hat, zu identifizieren. Sinnvollerweise können hier Rechnername, Firmenname oder ähnliches verwendet werden.

```
alert_mail_from = AppQueryr1@vectorsoft.de
alert_mail_to
```

Hier können ein oder mehrere Empfänger angegeben werden. Die einzelnen Empfänger-Adressen müssen durch Semikolon getrennt sein.

```
alert_mail_to = DbAdmin@vectorsoft.de;support@vectorsoft.de
alert_mail_to_sms
```

Hier können ein oder mehrere Empfänger angegeben werden. Die einzelnen Empfänger-Adressen müssen durch Semikolon getrennt sein. Die Empfänger erhalten eine E-Mail ohne Text, in der die letzte Zeile der Log-Datei im "Betreff" der Mail eingetragen ist. Diese E-Mails können zum Beispiel über einen SMS-Gateway in das Funknetz übermittelt werden. Sofern alert\_mail\_to\_sms definiert ist, kann auf alert\_mail\_to verzichtet werden.

```
alert_mail_to_sms='0172556611@d2-message.de'
alert_mail_proxy
```

Hier wird der Proxy angegeben, über den der Mailserver erreichbar ist. Die Angabe kann entfallen, wenn kein Proxy benötigt wird. Der Proxy wird in der Form <Proxy-Typ>:<IP-Adresse/Hostname>:<Portnummer> angegeben. <Proxy-Typ> gibt die Version des Proxies an. Es werden die Typen SOCKS Version 4 (SOCKSv4), 4a (SOCKSv4a) und 5 (SOCKSv5) unterstützt. Die Portnummer muss nur angegeben werden, wenn sie vom Standard (1080) abweicht.

```
alert_mail_proxy = SOCKSv4a:proxy.kunden.de:1080
alert_mail_tls
```

Dieser Parameter definiert, ob nach dem Verbindungsauflauf zum Mailserver eine verschlüsselte Verbindung (alert\_mail\_tls = Y) oder eine unverschlüsselte Verbindung (alert\_mail\_tls = N) verwendet werden soll. Unterstützt der Mailserver keine verschlüsselten Verbindungen, wird trotz alert\_mail\_tls = Y eine unverschlüsselte Verbindung hergestellt.

## Kontakt

```
alert_mail_tls = Y  
alert_lines
```

Die Größe der zu versendenden E-Mail kann über die maximale Anzahl der Zeilen beschränkt werden. Die Angabe der maximalen Zeilen ist optional. Wird der Eintrag nicht gesetzt oder der Wert 0 angegeben, werden bis zu 999 Zeilen aus der Log-Datei gesendet.

```
alert_lines = 20  
alert_interval
```

Nachdem eine E-Mail-Benachrichtigung gesendet wurde, wird die hier angegebene Zeitspanne gewartet, bis ein erneutes Senden einer Benachrichtigung möglich ist. Dadurch wird die Anzahl generierter Mails bei fortwährenden neuen Fehlereinträgen reduziert. Der Eintrag ist optional. Wird er nicht gesetzt, wird frühestens nach einer Stunde die LOG-Datei erneut gesendet. Es können Werte zwischen 5 Minuten (5m) und einem Tag (1d) angegeben werden.

```
alert_interval = 4h  
alert_test
```

Befindet sich in der Konfigurationsdatei der Eintrag alert\_test = 1, wird nach dem Starten des SOA-Service der Eintrag "Alert mail test" in die Protokolldatei geschrieben. Dieser Eintrag löst das Versenden der E-Mail aus. Auf diese Weise kann das Versenden der Mail getestet werden. Dieser Eintrag sollte nach einem erfolgreichen Test wieder entfernt werden, da sonst bei jedem Starten des Service-Clients eine Mail generiert wird.

```
alert_test = 1
```

## Minidump-Generierung

Die folgenden Eintragungen sind für SOA-Tasks mit graphischer Erweiterung (c16\_proc\_extended = Y) verfügbar, um im Fall eines Absturzes ein Minidump zu generieren. Bei SOA-Tasks ohne graphische Erweiterung werden die Optionen ignoriert, da bereits Minidump-Dateien generiert werden.

**apploptions**

Mit dieser Option kann das Anwendungsprotokoll für den erweiterten SOA-Task aktiviert werden. Dies ist notwendig, damit bei Abstürzen Minidump-Dateien erzeugt werden. Zum Aktivieren muss der Wert auf 4 gesetzt werden. Zum Deaktivieren, muss 0 eingetragen, oder die Option entfernt werden.

```
apploptions = 4  
applogpath
```

Mit der Option wird der Pfad für das Anwendungsprotokoll festgelegt, sofern apploptions gesetzt ist. Wird applogpath nicht definiert, wird das Anwendungsprotokoll im Verzeichnis der Konfigurationsdatei

## Kontakt

angelegt, bzw. geöffnet. Der Dateiname des Anwendungsprotokolles ist der Name des SOA-Task um \_dbg.lgb erweitert. Heisst der SOA-Task z. B. MyTask, dann ist der Name des Anwendungslogs MyTask\_dbg.lgb.

```
applogpath = C:\CONZEPT 16\Logs\
```

### Betriebsart TIME

Die folgenden Eintragungen sind nur für Services der Betriebsart TIME (siehe mode) notwendig.

```
time_delay_default
```

Dies ist die standardmäßige Zeitspanne die nach dem Aufruf der Ereignisprozedur vergeht, bevor das Ereignis erneut aufgerufen wird. Der Maximalwert beträgt 24 Stunden (1d), der Standardwert ist fünf Minuten. Die Zeit wird ab dem Beenden der Prozedur gemessen. Die Zeitspanne kann innerhalb der Ereignisprozedur über die Eigenschaft SvcCallDelay verändert werden.

```
time_delay_default = 5m
```

```
time_delay_error
```

Falls bei der Durchführung eines Ereignisses ein Fehler entsteht (keine Verbindung zur Datenbank, Laufzeitfehler oder ähnliches) ist dies die Zeitspanne, nach der erneut ein Ereignis ausgelöst wird. Es können Werte zwischen einer Minute (1m) und zwölf Stunden (12h) angegeben werden, der Standardwert ist eine Stunde.

```
time_delay_error = 1h
```

### Betriebsart SOCKET

Die folgenden Eintragungen sind nur für Services der Betriebsart SOCKET (siehe mode) notwendig.

```
socket_port *
```

Dies ist der TCP/IP-Port, auf dem der Dienst auf eingehende Verbindungen wartet. Der Port kann im Bereich von 1 bis 65535 angegeben werden. Die Portnummer darf auf dem System nicht bereits verwendet werden.

```
socket_port = 5601
```

```
socket_ip
```

Wenn der TCP/IP-Port nur auf bestimmten IP-Adressen verfügbar sein soll, können diese in diesem Parameter angegeben werden. Ohne Angabe dieser Einstellung ist der Port auf allen IP-Adressen des Rechners aktiv. Eine Beschränkung auf lokale Verbindungen kann beispielsweise mit der Angabe der Loopback-Adresse 127.0.0.1 erreicht werden. Mehrere IP-Adressen werden durch Semikolon getrennt angegeben.

```
socket_ip = 127.0.0.1;10.1.3.1
```

## Kontakt

Der Dienst kann auch nur für bestimmte Ports zur Verfügung gestellt werden. Der entsprechende Port kann durch einen Doppelpunkt getrennt nach der IP-Adresse angegeben werden. Ist kein Port angegeben, wird der Port in socket\_port verwendet.

```
socket_ip = 127.0.0.1:5050;10.1.3.1:443  
socket_timeout
```

Dieser Wert bestimmt die Zeitspanne für einen Timeout-Fehler beim Lesen oder Schreiben von Daten auf dem Socket. Der Wert kann auch in der Ereignisprozedur mit SckInfo() abgefragt oder verändert werden. Die zulässigen Werte liegen im Bereich von 3 Sekunden (3s) bis 30 Minuten (30m), der Standardwert ist 10 Sekunden. Bei Anwendungen im Internet sind höhere Werte (zum Beispiel 1 Minute) als im lokalen Netz (10 Sekunden) sinnvoll.

```
socket_timeout = 1m  
socket_keepalive
```

Beim Einsatz von Verbindungen, die auch nach der Ereignisprozedur noch offen bleiben (beispielsweise bei HTTP 1.1) muss eine maximale Zeitspanne eingestellt werden, in der neue Daten empfangen werden können (siehe Verarbeitungshinweise). Der Wert kann innerhalb einer Ereignisprozedur mit SckInfo() verändert werden. Die zulässigen Werte liegen im Bereich von 3 Sekunden (3s) bis 10 Minuten (10m), der Standardwert ist 20 Sekunden. Der Wert ist unabhängig von socket\_timeout.

```
socket_keepalive = 20s  
ip_connection_limit
```

Diese Einstellung beschränkt die Anzahl von gleichzeitigen Verbindungen, die von einer einzelnen Quell-IP-Adresse aufgebaut werden können. Dies dient hauptsächlich dem Schutz vor einem Überlauf der Verbindungen durch Anwendungs- oder Programmfehler der Gegenstelle. Standardmäßig ist keine Beschränkung aktiv, zulässige Werte liegen im Bereich von 0 bis 1000.

```
ip_connection_limit = 5
```

## Verschlüsselte Kommunikation über SOCKET-basierte Tasks

Ein socket-basierter SOA-Service kann optional mit Verschlüsselung unter Verwendung der Transport Layer Security betrieben werden. Dafür ist ein entsprechendes SSL/TLS-Zertifikat notwendig.

Für die Verschlüsselung sind vier zusätzliche Einträge in der Konfigurationsdatei des Tasks notwendig:

```
tls_certificate_file
```

Hier wird der vollständige Pfad- und Dateiname des Zertifikatsdatei

## Kontakt

angegeben. Die Datei muss im PEM-Format vorliegen.

```
tls_certificate_file = "c:\certificate\soa\soacertificate.pem"
```

In der Datei muss das Zertifikat Base64-kodiert gespeichert und von -----BEGIN CERTIFICATE----- / -----END CERTIFICATE----- eingeschlossen sein. In der Datei können mehrere Zertifikate einer Zertifikatskette vorhanden sein. Die Zertifikate müssen in der Reihenfolge <mein Zertifikat><Zertifikat des Ausstellers>...<Root-Zertifikat> angegeben werden.

**tls\_private\_key\_file**

Hier wird der vollständige Pfad- und Dateiname der Datei angegeben, die den zum Zertifikat passenden privaten Schlüssel enthält. Die Datei muss im PEM-Format vorliegen und einen RSA-Schlüssel enthalten. Falls der private Schlüssel in der Zertifikatsdatei enthalten ist, wird hier der gleiche Dateiname wie bei **tls\_certificate\_file** eingetragen. Es spielt dabei keine Rolle, ob der Schlüssel am Anfang oder am Ende der Datei ist.

```
tls_private_key_file = "c:\certificate\soa\soaprivatekey.pem"
```

**tls\_private\_key\_password**

Dies ist das Zugangskennwort des privaten Schlüssels.

```
tls_private_key_password = rosebud
```



Das Passwort darf nicht von Anführungszeichen ("") eingeschlossen werden.

**tls\_security\_level**

Dies ist der Mindestsicherheitslevel von aufgebauten Verbindungen. Folgende Werte können angegeben werden: Med / 1, Hi / 2 und Max / 3. Ist keine Option angegeben, wird Med verwendet. Die Mindestsicherheitslevel entsprechen den Konstanten SckTlsMed, SckTlsHigh und SckTlsMax für SckConnect().

```
tls_security_level = Med
```

**Betriebsart DRIVE**

Die folgenden Eintragungen sind nur für Services der Betriebsart DRIVE (siehe mode) notwendig.

**c16\_server \***

In diesem Eintrag wird der CONZEPT 16-Server eingetragen. Der Eintrag erfolgt in Form der IP-Adresse oder des IP-Namens des Serverrechners. Zur Verbindung wird immer das Protokoll TCP/IP verwendet. Bei Verwendung der Hot-Standby-Option wird nur der Primärserver angegeben.

```
c16_server = dbprimserv
```

## Kontakt

### c16\_database

In diesem Eintrag wird die Datenbank eingetragen. Der Eintrag entspricht dem Namen der Datenbank, wie sie beim CONZEPT 16-Server eingetragen ist. Dieser Eintrag wird in der Betriebsart DRIVE nur im Hot-Standby-Betrieb benötigt.

```
c16_database = codelibrary  
socket_port *
```

Dies ist der TCP/IP-Port, an den der Laufwerkstreiber seine Datenpakete sendet. Der Port kann im Bereich von 1 bis 65535 angegeben werden. Die Portnummer darf auf dem System nicht bereits verwendet werden. Die Applikation, die die Datenpakete verarbeitet, muss eine Verbindung auf diesen Port erwarten.

```
socket_port = 50001  
socket_timeout
```

Dieser Wert bestimmt die Zeitspanne für einen Timeout-Fehler beim Lesen oder Schreiben von Daten auf dem Socket. Die zulässigen Werte liegen im Bereich von 5 Sekunden (5s) bis 30 Minuten (30m), der Standardwert ist 1 Minute.

```
socket_timeout = 1m
```

Der Laufwerkstreiber arbeitet immer mit einem SOA-Task vom Typ SOCKET zusammen. Dieser Task muss auf den gleichen Port eingestellt sein, wie der Laufwerkstreiber. Der Task sollte die Verwendung von Oberflächenobjekten (c16\_proc\_extended = N) nicht unterstützen.

## Kontakt

### Beispiel - c16\_soa.cfg

Beispiel für die Konfigurationsdatei des SOA-Service

Die Konfigurationsdatei muss für die Anwendung angepasst werden. Die hier verwendeten Werte können für die ersten Schritte bei der Entwicklung eines SOA-Service genutzt werden. Details zu den verschiedenen Eintragungen sind in dem Abschnitt Konfiguration des SOA-Service erläutert.

Der Task benötigt eine eigene Konfigurationsdatei <Name des Task>.cfg. Eine Beispiel-Datei befindet sich im Abschnitt Beispiel - SOA-Task.cfg.

```
; --- Name des Tasks[MobileApp]; --- Beschreibung;      Optionaldescription      = "CodeLibrary" -
```

## Kontakt

### Beispiel - SOA-Task.cfg

Beispiel für die Konfigurationsdatei eines SOA-Tasks

Die Konfigurationsdatei muss für die Anwendung angepasst werden. Die hier verwendeten Werte können für die ersten Schritte bei der Entwicklung eines SOA-Service-Task genutzt werden. Details zu den verschiedenen Eintragungen sind in dem Abschnitt Konfiguration des SOA-Service erläutert.

```
---; --- Allgemeine Parameter; ---; --- IP-Adresse/Host-Name des Serversc16_server  
---; --- Alarmierungsparameter (Optional); ---; --- IP-Adresse/Host-Name des E-Mail-Servers; al  
---; --- Session-spezifische Parameter (Optional); ---; --- Sitzungslimit (0 = unbegrenzt);
```

## Kontakt

### SOA-Service - Log-Datei

Einträge in den Log-Dateien des SOA-Service

Bei der Verwendung des SOA-Service werden verschiedene Log-Dateien im Verzeichnis der Konfigurationsdateien angelegt. Der SOA-Service selbst schreibt die Datei c16\_soa.lgb. Für jeden Task wird eine weitere Datei bestehend aus dem Namen des Task mit der Erweiterung .lgb erstellt.

Die Einträge in den Log-Dateien sind im Abschnitt Log-Einträge beschrieben.

## Kontakt

### Ereignisfunktion des SOA-Service

Funktion, die durch einen Task des SOA-Service aufgerufen wird

Name	Typ	Beschreibung
<u>aObjHdl</u>	<u>handle</u>	Deskriptor des Task-Objekts
<u>aEvtType</u>	int	Ereignis-Typ <u>SOA-Service</u> , <u>c16 procedure</u> , <u>Task</u> ,

Siehe

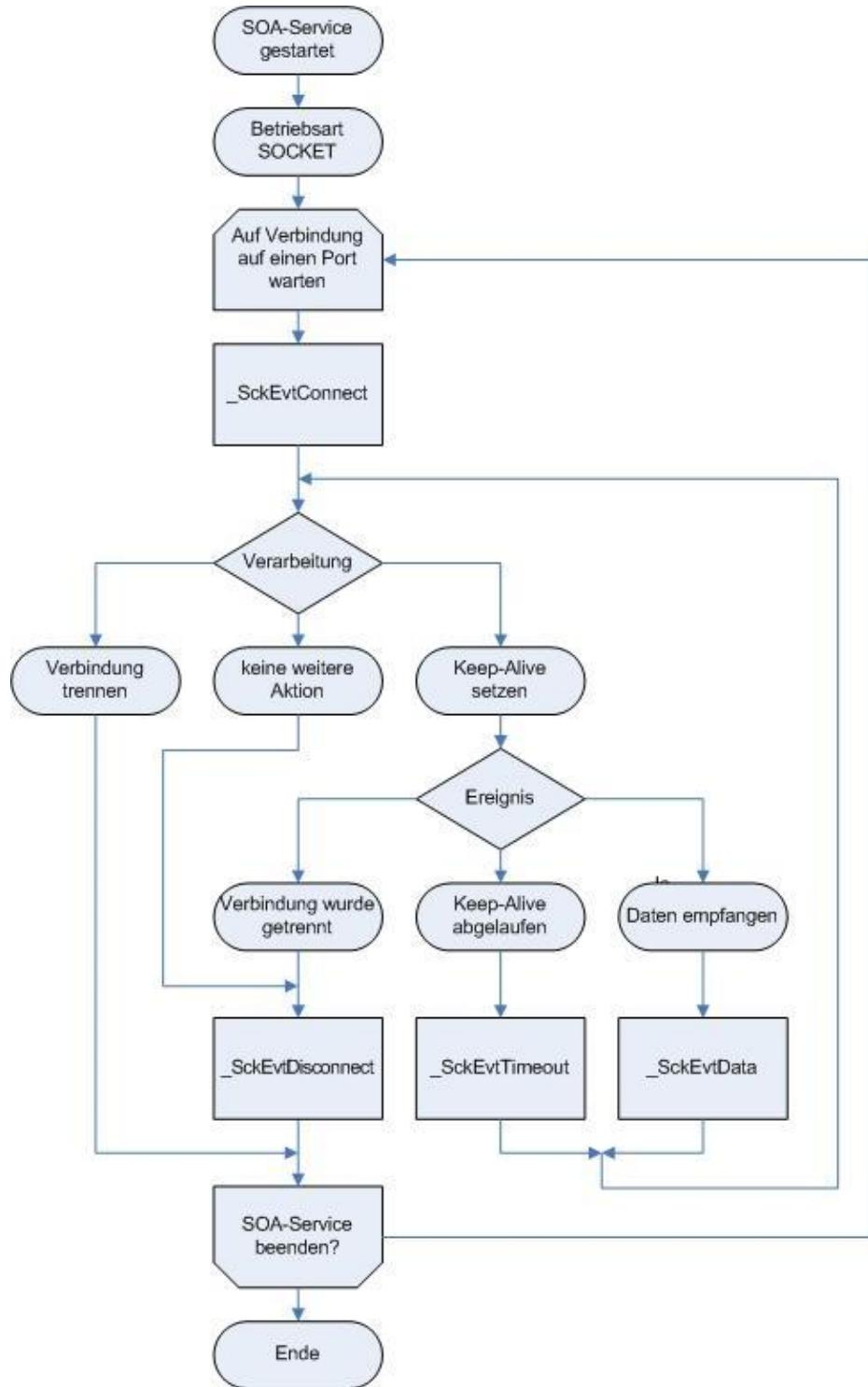
#### Job

Diese Funktion wird je nach Betriebsart des Task des SOA-Service zu unterschiedlichen Zeitpunkten aufgerufen. Arbeitet der Task in der Betriebsart TIME (siehe mode), erfolgt der Aufruf beim Start des SOA-Service und anschließend in regelmäßigen Abständen. Die Zeitabstände sind in der Einstellung time delay default bzw. im Task-Objekt angegeben. Bei der Betriebsart SOCKET erfolgt der Aufruf erst bei einem Verbindungsauftbau über den Socket.

Ereignisaufrufe                  Ereignisaufrufe

Betriebsart TIME Betriebsart SOCKET

## Kontakt



Damit die Funktion aufgerufen werden kann, muss sie in dem Eintrag c16 procedure eingetragen werden. Die Funktion selbst muss mit folgendem Funktionskopf definiert sein:

```
sub <Name>( aObjHdl : handle; // Task-object aEvtType : int; // Event-type)
```

**aObjHdl**

## Kontakt

In dem Parameter aObjHdl wird der Deskriptor des verwendeten Task-Objekt übergeben. Jedes Ereignis hat ein eigenes Objekt, welches über eine Reihe von Eigenschaften verfügt. Der Typ des Deskriptors ist in allen Fällen HdlSvcTime oder HdlSvcSocket. Der Untertyp ist abhängig davon, ob der Job als Thread (JobThread) oder als Prozess (JobProcess) gestartet wurde.

### aEvtType

In aEvtType wird der Ereignistyp übergeben. Beim TIME-Service gibt es nur einen Typ von Ereignis, aEvtType enthält daher immer Null. Beim SOCKET-Service gibt es vier unterschiedliche Ereignistypen.

#### SckEvtConnect

Bei einer neuen Verbindung wird dieses Ereignis ausgelöst. Falls die gesamte Verarbeitung in der Ereignisfunktion durchgeführt werden kann, sollte der Socket am Ende mit JobClose() geschlossen werden (Eigenschaft SvcSckHandle des übergebenen Objekts). Damit ist die Verbindung beendet und es gibt auch keine weiteren Ereignisse mehr für diese Verbindung. Um den Socket für einen weiteren Datenaustausch zu verwenden, muss mit Keep-Alives gearbeitet werden (siehe Verarbeitungshinweise).

#### SckEvtDisconnect

Falls der Socket der Verbindung nicht aktiv mit SckClose() geschlossen wurde, wird nach dem Trennen der Verbindung in jedem Fall dieses Ereignis aufgerufen. Der Socket ist beim Aufruf der Ereignisprozedur bereits geschlossen.

#### SckEvtData

Bleibt der Socket nach dem Ende der Ereignisprozedur noch geöffnet, kann dieses Ereignis auftreten, wenn innerhalb der Keep-Alive-Zeit neue Daten empfangen werden.

#### SckEvtTimeout

Bleibt der Socket nach dem Ende der Ereignisprozedur noch geöffnet, kann dieses Ereignis auftreten, wenn innerhalb der Keep-Alive-Zeit keine neuen Daten mehr empfangen wurden.

Eine Funktion mit identischem Funktionskopf muss angegeben werden, wenn mit der Anweisung JobStart() ein Job gestartet wird. Beim Aufruf der Ereignisfunktion wird dann ein Job-Objekt übergeben. Der Typ des Deskriptors ist in allen Fällen HdlJob, wobei der Untertyp JobThread oder JobProcess sein kann.



Innerhalb des SOA-Service können nur A+ Prozeduren verarbeitet werden. Der Aufruf von Prozeduren mit dem Befehl CallOld() wird ignoriert.

## Kontakt

### SOA-Service - Administration des SOA-Service

Beschreibung der Administration des SOA-Service Siehe

#### Blog

Die Administration des SOA-Service erfolgt über das Control-Center oder über ein eigenes Script-Utility. Im folgenden werden die Unterschiede zum Script-Utility des Servers beschrieben.

Das Script-Utility ist ein Kommandozeilenprogramm, welches den Status des Service und seinen Tasks abfragen und auch setzen kann. Daneben kann es auch zur Anzeige der binären Log-Dateien verwendet werden. Das Script-Utility ist dabei für folgende Betriebssysteme verfügbar:

**c16\_soa\_cmd\_win.exe** Windows 32- und 64-Bit

**c16\_soa\_cmd\_l32** Linux 32-Bit

**c16\_soa\_cmd\_l64** Linux 64-Bit

Dem Programm müssen zur Durchführung der Kommandos bestimmte Argumente an der Kommandozeile übergeben werden. Die Argumente hängen vom jeweiligen Kommando ab.

```
c16_soa_cmd_* <Kommando> [<Argumente>] [-[no]quiet[=inf|err]]
```

Die allgemeine Kommandozeile ist identisch zum Server Script-Utility.

Der Rückgabewert gibt Aufschluss darüber, ob und mit welchem Ergebnis das Kommando durchgeführt wurde. Wird das Script-Utility in einer Script-Datei verwendet, kann dieser Rückgabewert über den Errorlevel des Betriebssystems ermittelt und ausgewertet werden. Zusätzlich zum Rückgabewert wird auf der Kommandozeile auch ein Fehlerwert und eine entsprechende Information ausgegeben. Die ausgegebenen Fehlertexte entsprechen den Log-Einträgen.

Das SOA Script-Utility unterstützt die folgenden Kommandos:

#### help

Hilfe zum Script-Utility oder einem bestimmten Kommando anzeigen

#### version

Versionsinformationen des Script-Utility anzeigen

#### status

Status-Informationen des Task anzeigen

#### start

Task starten

#### stop

Task beenden

#### receive Log-Dateien

abholen

## **Kontakt**

**decode**

**Anzeige des Inhalts von binären Log-Dateien**

**list**

**Auflistung der konfigurierten Tasks**

## Kontakt

### Script-Utility (SOA-Service)

decode

Siehe [Script-Utility  
\(SOA-Service\)](#)

decode - Anzeige des Inhalts von binären Log-Dateien

Syntax:

```
c16_soa_cmd_win.exe decode <blog1> [<blog2>] [-start=<date>] [-end=<date>] [-days=<int>] [-record
```

Die Kommando-Zeile entspricht der des Server [Script-Utilities](#).

Rückgabe:

- 1 - Kommando erfolgreich ausgeführt.
- 2 - Fehler aufgetreten.

## Kontakt

### Script-Utility (SOA-Service)

**help**

Siehe [Script-Utility  
\(SOA-Service\)](#)

**help** - Hilfe zum Script-Utility oder einem bestimmten Kommando anzeigen

**Syntax:**

```
c16_soa_cmd_win.exe help [<Kommando>]
```

Ohne Angabe des <Kommando> wird eine Übersicht aller vorhandenen Kommandos angezeigt. Mit Angabe des <Kommando> wird die Befehlszeile und eine Beschreibung des angegebenen Kommandos angezeigt.

## Kontakt

### Script-Utility (SOA-Service)

list

Siehe [Script-Utility  
\(SOA-Service\)](#)

list - Tasks auflisten

Syntax:

```
c16_soa_cmd_win.exe list [-service=<address>]
```

Die Liste der konfigurierten Tasks wird ausgegeben.

Optional kann die Liste der Tasks von einem anderen SOA-Service ermittelt werden, dazu wird die IP-Adresse des SOA-Services mit dem Argument -service angegeben.

Rückgabe:

1 - Taskliste ermittelt.

2 - Fehler aufgetreten.

## Kontakt

### Script-Utility (SOA-Service)

receive

Siehe [Script-Utility  
\(SOA-Service\)](#)

receive - Log-Dateien abholen

Syntax:

```
c16_soa_cmd_win.exe receive <path> [-task=<string>] [-service=<address>] [-options=<string>] [-ty
```

Das Kommando holt die Logdateien des Service oder eines Task vom SOA-Service ab.

Argumente:

<path>	Zielpfad für die Log-Dateien.
[-task=<string>]	Name des Tasks. Entfällt, wenn nur die Log-Dateien vom Service abgeholt werden.
[-service=<address>]	Adresse des SOA-Services. Wird dieser Parameter nicht angegeben, wird der lokale SOA-Service verwendet.
[-type=<string>]	Log-Dateien, die angefordert werden sollen. Die einzelnen Typen können kombiniert werden. Wird kein Parameter angegeben, wird automatisch die Log-Datei des Services übertragen. s Log-Datei des Services t Log-Datei des Tasks
[-options=<string>]	Zusätzliche Optionen y Bei Angabe dieser Option werden schon vorhandene Log-Dateien überschrieben. Dies ist ansonsten nicht der Fall. x Bei Verwendung dieser Option wird die gesamte Übertragung bei einem Übertragungsproblem abgebrochen. Ansonsten wird einfach mit der nächsten Log-Datei fortgefahren

Rückgabe:

1 - Die Log-Datei(en) wurde(n) korrekt übertragen.

2 - Fehler aufgetreten.

## Kontakt

### Script-Utility (SOA-Service)

start

Siehe [Script-Utility  
\(SOA-Service\)](#)

start - Task starten

Syntax:

```
c16_soa_cmd_win.exe start <taskname> [-service=<address>]
```

Der angegebene Task wird gestartet. Das Kommando wartet bis der Task gestartet ist oder das Timeout (30 Sekunden) überschritten wurde.

Optional kann der Task von einem anderen SOA-Service gestartet werden, dazu wird die IP-Adresse des SOA-Services mit dem Argument -service angegeben.

Rückgabe:

- 1 - Task gestartet.
- 2 - Fehler aufgetreten.

## Kontakt

### Script-Utility (SOA-Service)

status

Siehe [Script-Utility  
\(SOA-Service\)](#)

status - Status-Informationen des Task anzeigen

Syntax:

```
c16_soa_cmd_win.exe status <taskname> [-service=<address>]
```

Der Status des angegebenen Task wird angezeigt.

Task is running Task ist gestartet.

Task is stopped Task ist gestoppt.

Optional kann der Status eines Tasks von einem anderen SOA-Service abgefragt werden, dazu wird die IP-Adresse des SOA-Services mit dem Argument -service angegeben.

Rückgabe:

0 - Task läuft nicht.

1 - Task läuft.

2 - Fehler aufgetreten.

## Kontakt

### Script-Utility (SOA-Service)

**stop**

Siehe [Script-Utility  
\(SOA-Service\)](#)

**stop - Task beenden**

**Syntax:**

```
c16_soa_cmd_win.exe stop <taskname> [-service=<address>]
```

**Der angegebene Task wird gestoppt. Das Kommando wartet bis der Task gestoppt oder das Timeout (30 Sekunden) überschritten wurde.**

**Optional kann der Task von einem anderen SOA-Service gestoppt werden, dazu wird die IP-Adresse des SOA-Services mit dem Argument -service angegeben.**

**Rückgabe:**

- 1 - Task gestoppt.**
- 2 - Fehler aufgetreten.**

## Kontakt

### Script-Utility (SOA-Service)

**version**

Siehe [Script-Utility  
\(SOA-Service\)](#)

**version** - Versionsinformationen des Script-Utility anzeigen

**Syntax:**

```
c16_soa_cmd_win.exe version
```

**Anzeige der Version des Script-Utility.**

## Kontakt

Befehle des SOA-Service

Befehle zur Programmierung innerhalb des SOA-Service Liste

sortiert

nach

Gruppen,

Alphabetische

Liste aller

Die Befehle werden im Zusammenhang mit dem SOA-Service verwendet.

Befehle

JobClose

JobControl

JobOpen

JobSleep

JobStart

SvcLog

SvcSessionControl

## Kontakt

obj -> JobClose()

—————  
—————

**Kontrollobjekt entfernen**  
Deskriptor eines  
obj

**Kontroll-Objekts**

**Verwandte**

Siehe **Befehle**,

**JobOpen()**

Die Anweisung kann innerhalb einer **Ereignisfunktion des SOA-Service**, sowie im **Standard-** oder **Advanced-Client** ausgeführt werden.

Die Funktion schließt das **JobControl**-Objekt mit dem Deskriptor (obj). Der Job kann in der selben Prozedur oder in einer anderen Ereignisprozedur des Tasks-Prozesses erneut mit **JobOpen()** geöffnet werden, sofern der Job noch aktiv ist oder die beim Starten des Jobs angegebene Zeitspanne nach Jobende noch nicht verstrichen ist.

Mögliche Laufzeitfehler:

**ErrHdlInvalid** Der übergebene Deskriptor ist ungültig.

## Kontakt

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



Job kontrollieren

obj	Deskriptor des JobControl-Objekts
<b>Durchzuführende Funktion</b>	
<u><a href="#">JobWake up</a></u>	Suspendierten
<u><a href="#">JobStop</a></u>	Job aktivieren Job beenden, Ende nicht abwarten
<u><a href="#">JobTerminate</a></u>	Job beenden, Ende abwarten
<u><a href="#">JobMsxTimeoutRead</a></u>	Timeout abfragen / setzen
<u><a href="#">JobEventReceiver</a></u>	Frame-Deskriptor für Job-Events setzen bzw. entfernen
int1	neuer Wert des Timeouts /

int2  
Frame-Deskriptor  
Resultat [int](#) **Ergebnis Text** der Abfrage (siehe  
Siehe [Inter-Thread-Kommunikation](#)  
[\(Blog\)](#)

Die Anweisung kann innerhalb einer Ereignisfunktion des SOA-Service, sowie im Standard-  
oder Advanced-Client ausgeführt werden.

Mit dieser Funktion kann ein JobControl- oder Job-Objekt verschiedene Funktionen durchführen.  
In (obj) wird der Deskriptor des Objektes (siehe JobOpen()) übergeben. In (int1) steht der Typ der  
durchzuführenden Funktion.

Für JobControl-Objekte sind folgende Optionen verwendbar:

### [JobWake up](#)

Mit dieser Option wird der Job aktiviert, wenn er sich in der eigenen Ereignisfunktion mit der  
Anweisung JobSleep() suspendiert hat. Der Rückgabewert ist ErrTerminated, wenn der Job  
bereits beendet ist. Ansonsten ist das Resultat ErrOk.

### [JobStop](#)

Diese Funktion setzt die Eigenschaft StopRequest für den Job ohne auf das Ende des  
Jobs zu warten. Der Rückgabewert ist immer ErrOk.

### [JobTerminate](#)

Diese Funktion setzt die Eigenschaft StopRequest für den Job und wartet darauf, dass  
sich der Job beendet. Der Rückgabewert ist immer ErrOk.

## Kontakt

### JobMsxTimeoutRead

Mit dieser Option kann der Timeout für MsxRead() auf die Message-Pipeline abgefragt (zwei Argumente) oder gesetzt werden (drei Argumente, der neue Wert steht in (int2)). Das Resultat ist der aktuelle bzw. neue Wert des Timeouts in Millisekunden (siehe Verarbeitungshinweise zum SOA-Service).

Für Job-Objekte sind folgende Optionen verwendbar:

### JobMsxTimeoutRead

Mit dieser Option kann der Timeout für MsxRead() auf die Message-Pipeline abgefragt (zwei Argumente) oder gesetzt werden (drei Argumente, der neue Wert steht in (int2)). Das Resultat ist der aktuelle bzw. neue Wert des Timeouts in Millisekunden (siehe Verarbeitungshinweise zum SOA-Service).

### JobEventReceiver

Mit dieser Option kann ein Frame-Deskriptor gesetzt werden (drei Argumente, der Deskriptor steht in (int2)), der durch JobEvent() ausgelöste EvtJob-Ereignisse empfängt. Bei Übergabe von nur zwei Argumenten wird ein zuvor gesetzter Deskriptor wieder entfernt.

Mögliche Laufzeitfehler:

ErrHdlInvalid Der in (obj) angegebene Deskriptor ist ungültig.

ErrValueInvalid Der in (int1) oder (int2) übergebene Wert ist nicht gültig.

## Kontakt

JobOpen(int1[,



handle2]) : handle

Kontrollobjekt erzeugen

int1 Id eines Job-Objekts

handle2 Deskriptor eines Dialogs

Deskriptor des

Resultat handle JobControl-Objekts

oder Fehlerwert

Verwandte Befehle,  
JobControl, JobClose(),

Siehe

Inter-Thread-Kommunikation

(Blog)

Die Anweisung kann innerhalb einer Ereignisfunktion des SOA-Service, sowie im Standard- oder Advanced-Client ausgeführt werden.

Mit diesem Befehl wird für den Job mit der Id (int1) ein Kontrollobjekt angelegt. Pro Job kann nur ein Kontrollobjekt vorhanden sein. Das Resultat ist ErrInUse, wenn bereits ein Kontrollobjekt für diesen Job besteht. Existiert kein Job mit der angegebenen Id, wird ErrUnknown zurückgeliefert. Das Kontrollobjekt sollte vor Ende der Prozedur mit JobClose() wieder geschlossen werden.

Im Parameter (handle2) kann optional der Deskriptor eines Dialoges angegeben werden. Ereignisse, die mit JobEvent() von dem Job ausgeführt werden, werden an das Ereignis EvtJob von dem angegebenen Dialog weitergeleitet.



Der Parameter (handle2) kann nur beim Aufruf aus dem Standard- oder Advanced-Client angegeben werden.

## Kontakt

**JobSleep(int1) :**



**logic**

**Job suspendieren**

Wartezeit in

int1

Millisekunden

**Resultat logic Wartezustand abgebrochen**

Verwandte Befehle,

Siehe

JobControl()

Die Anweisung kann innerhalb eines Jobs ausgeführt werden. Jobs können durch eine Ereignisfunktion des SOA-Service, einen Standard- oder Advanced-Client gestartet werden.

Diese Funktion hält die Verarbeitung im Job für (int1) Millisekunden an. Im Unterschied zu SysSleep() kann der Wartezustand abgebrochen werden. In diesem Fall wird als Resultat true zurückgegeben. Ist die Wartezeit ohne Unterbrechung verstrichen, so ist das Resultat false. Ein Abbruch findet statt, wenn der komplette Task gestoppt werden soll oder wenn der Job durch JobControl(..., JobWakeUp) aufgeweckt wird.

Mögliche Laufzeitfehler:

ErrValueInvalid In (int1) wurde ein Wert < 0 angegeben.

## Kontakt

JobStart(int1, int2,  
alpha3[, alpha4[,  
alpha5[, handle6]]]) : 

int

Job starten

Ausführungstyp:

JobThread Thread innerhalb des

Prozesses

JobProcess Betriebssystemprozess

JobCopyBuffers Inhalt der globalen  
Feldpuffer kopieren (nur

int1 vom Client aus)

JobWinPrt Verwendung von

Oberflächenobjektbefehlen

ermöglichen (nur vom

Client aus)

JobStartNoRTE Laufzeitfehler bei JobStart

unterbinden

Zeitspanne, in der die

int2 Job-Informationen zur Verfügung  
stehen

alpha3 Name der Ereignisfunktion

alpha4 Argumente für den Job (optional)

alpha5 Beschreibung des Jobs (optional)

handle6 Socket-Deskriptor (optional)

Job-Id oder Fehlerwert

Wert > 0 Job-Id des  
neuen Jobs

Resultat int ErrTerminated Job  
erfolgreich  
durchgeführt

anderer Wert < Fehlerwert

0

Siehe Verwandte Befehle, JobOpen()

Die Anweisung kann innerhalb einer Ereignisfunktion des SOA-Service, sowie im Standard-  
oder Advanced-Client ausgeführt werden.

Mit diesem Befehl wird ein neuer Job gestartet. In (int1) wird der Ausführungstyp des Jobs  
angegeben: JobThread startet einen neuen Job-Thread innerhalb des aktuellen  
Betriebssystemprozesses, JobProcess einen neuen Betriebssystemprozess für diesen Job (siehe  
Verarbeitungshinweise).



Der Ausführungstyp JobProcess steht nur im SOA-Service zur Verfügung. Wird sie im  
Standard- oder Advanced-Client angegeben, wird der Laufzeitfehler ErrValueInvalid  
ausgelöst.

Der Ausführungstyp JobThread kann im Standard- oder Advanced-Client mit der Optionen  
JobCopyBuffers und JobWinPrt kombiniert werden.

## Kontakt

Ist die Option JobCopyBuffers angegeben, werden beim Start des Jobs die aktuellen Feldpuffer des Clients in die Feldpuffer des Jobs übertragen.



Wurde zwischen Anmelden des Benutzers an der Datenbank und dem JobStart(\_JobThread | \_JobCopyBuffers, ...) die Datenstruktur neu aufgebaut, wird der Job nicht gestartet und die Eigenschaft JobErrorCode auf den Fehler ErrIllegalOp gesetzt.

Die Option JobWinPrt ermöglicht die Verwendung von Dialogen und Oberflächenobjektbefehlen im Job.



Ein mit JobStart() gestarteter Thread ist von anderen Threads isoliert. Somit kann nur auf Eigenschaften und Ereignisse von Oberflächenobjekten des eigenen Threads zugegriffen werden. Zur Thread-Kommunikation können die Message-Exchange-Befehle verwendet werden. Zum Signalisieren von Nachrichten kann der Befehl JobEvent() verwendet werden.



Die Option JobWinPrt sollte nur verwendet werden, wenn auch auf Oberflächenobjektbefehle zugegriffen wird.

In (int2) wird die Zeitspanne in Sekunden angegeben, in der der Job noch nach seiner Beendigung mit JobOpen() geöffnet werden kann. Bei einem Wert von 0 kann der Job geöffnet werden, solange er ausgeführt wird. Der Maximalwert für (int2) beträgt 86400 Sekunden (24 Stunden).

In (alpha3) befindet sich der Name der Ereignisfunktion, die der Job aufrufen soll. In (alpha4) kann optional ein Argumenttext übergegeben werden, der in der Jobprozedur über die Eigenschaft JobData abgerufen werden kann. In (alpha5) kann optional ein Beschreibungstext für den Job angegeben werden, der in der Jobprozedur über die Eigenschaft SvcDescription abgerufen werden kann. Die Argumente und die Beschreibung können jeweils bis zu 8192 Zeichen lang sein.

Optional kann in (handle6) ein Socket-Deskriptor (siehe SckConnect()) angegeben werden. Dabei wird eine Kopie des Deskriptors erstellt, der auf die gleiche Socket-Verbindung verweist. Diese ist in der Job-Funktion über die Eigenschaft JobSckHandle zugreifbar. Die Socket-Verbindung kann dann sowohl in dem Job, als auch in der aktuellen Prozedur verwendet werden. Somit kann beispielsweise im Job auf neue Nachrichten gewartet und gleichzeitig in der aktuellen Prozedur Nachrichten versendet werden. Hierbei sollte jedoch eine Seite nur lesen und eine nur schreiben.

Die Socket-Verbindung muss nur auf der Seite geschlossen werden, die JobStart() aufgerufen hat. Wird der Socket auf der Seite des Jobs geschlossen, wird die Verbindung auf beiden Seiten getrennt. Jedoch bleibt der Socket-Deskriptor in der Prozedur, die den Job gestartet hat, erhalten. Dieser muss dann separat mit SckClose() geschlossen werden. Am Ende des Jobs wird die erzeugte Kopie des Socket-Dekriptors entfernt. Die Socket-Verbindung bleibt am Ende des Jobs erhalten und kann in der aufrufenden Prozedur bis zum SckClose() verwendet werden.



Es kann kein Socket-Deskriptor angegeben werden, der mit einer der \_SckTls...- Optionen erzeugt wurde.

Der Rückgabewert ist die Job-Id des neuen Jobs (Wert > 0). Ist der Job bereits durchgeführt, bevor JobStart() zurück gekommen ist, oder soll der Client, der Soa-Service oder der aktuelle Job beendet werden, wird ErrTerminated

## Kontakt

zurückgegeben. Falls beim Start des Jobs ein Fehler aufgetreten ist, enthält der Rückgabewert den Fehlerwert (Wert < 0). Die Job-Id ist innerhalb des laufenden Tasks eindeutig. Sie wird zum Erzeugen eines Kontroll-Objekts mit JobOpen() benötigt.



Ist die Option (int1) JobStartNoRTE angegeben, werden keine Laufzeitfehler ausgelöst.  
Statt dessen wird der Wert des Laufzeitfehlers als Fehlerwert zurückgeliefert.

Mögliche Laufzeitfehler:

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

ErrNoProcInfo Prozedur ist nicht vorhanden oder wurde nicht übersetzt

ErrNoSub Prozedurfunktion ist nicht vorhanden

## Kontakt

SvcLog(int1, logic2, alpha3)  Benutzerlog des  
SOA-Service schreiben

Klassifizierung der Information:

LogInfo Informativer

int1 Eintrag

.LogWarning Warnungseintrag

.LogError Fehlereintrag

logic2 Systemeintrag

alpha3 Text des Eintrags (max. 250

Zeichen)

Verwandte Befehle,

Siehe [Log-Einträge des SOA-Service](#)

Die Anweisung kann nur in einer [Ereignisfunktion des SOA-Service](#) durchgeführt werden.

Mit diesem Befehl wird ein Benutzerlog für den [SOA-Service](#) geschrieben.

Die Einträge werden in drei Klassen unterschieden:

LogInfo

Diese Einträge sollten nur informativen Charakter besitzen.

.LogWarning

Der Eintrag kennzeichnet einen Warnzustand.

.LogError

Der Eintrag kennzeichnet einen Fehlerzustand.

Die unterschiedlichen Klassen werden bei der Anzeige des Protokolls mit dem [Log-Viewer](#) durch entsprechende Symbole gekennzeichnet. Neben den übergebenen Parametern wird in der Log-Datei das Datum und die Uhrzeit festgehalten. Mit dem Parameter (logic2) wird definiert, ob es sich bei dem Eintrag um einen Systemeintrag (true) handelt, oder nicht (false). Systemeinträgen wird in der Log-Datei eine höhere Priorität bei der Langzeitspeicherung gewährt (siehe [Benutzerlogs](#)). Die Benutzer-ID wird nur dann eingetragen, wenn es sich bei dem Log-Eintrag nicht um einen System-Eintrag (logic2 = false) handelt.

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

Mögliche Laufzeitfehler:

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

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

## Kontakt

SvcSessionControl(int1[, int2]) : int



Verwaltung der Sitzungsinformationen

Befehlsmodus

\_SvcSessionCreate Sitzung  
anlegen

int1      \_SvcSessionLoad      Sitzung  
              lesen  
          \_SvcSessionDelete Sitzung  
              löschen

int2      Session-ID (optional)

Resultat int Session-ID oder Fehlerwert

Siehe      Verwandte Befehle

Die Anweisung kann nur innerhalb einer Ereignisfunktion des SOA-Service ausgeführt werden.

Mit dieser Funktion werden die Sessions innerhalb der Verarbeitung des SOA-Services verwaltet. In (int1) wird der Befehlsmodus übergeben. Folgende Modi stehen zur Verfügung:

SvcSessionCreate

SvcSessionLoad

SvcSessionDelete

\_SvcSessionCreate - Sitzung erzeugen

Mit der Anweisung SvcSessionControl(\_SvcSessionCreate) wird eine neue Session angelegt und der laufenden Prozedur zugeordnet. Der Rückgabewert ist die Session-ID (Wert größer 0) oder der Fehlerwert (Wert kleiner 0). Nach dem Ende der laufenden Prozedur werden die vorhandenen globalen Datenbereiche und Deskriptoren unter dieser Session-Id automatisch gespeichert.

Deskriptoren von Datensatzpuffern, Oberflächen- und Druckobjekten werden dabei nicht gesichert, sondern gelöscht.

Die Session-ID kann beispielsweise im HTTP-Protokoll als Cookie übertragen werden.

Im Task-Objekt ist die Session-ID als Eigenschaft SvcSessionID verfügbar.

Folgende Fehlerresultate sind möglich:

ErrSvcSessionState Der Prozedur ist bereits eine Session zugeordnet.

ErrLimitExceeded Die maximale Anzahl von Sessions ist erreicht. \_SvcSessionLoad

- Sitzung lesen

Mit der Anweisung SvcSessionControl(\_SessionLoad, <SessionID>) werden die gesicherten Deskriptoren und Datenbereiche einer Session in die Prozedurumgebung übertragen. Die Session-ID muss im Argument (int2) übergeben werden. Die Session wird dabei der laufenden Prozedur zugeordnet, wodurch nach dem Ende der Prozedur die Daten auch wieder unter dieser Session-ID gesichert werden.

## Kontakt

Folgende Fehlerresultate sind möglich:

**ErrSvcSessionState** Die Prozedur hat bereits eine zugeordnete Session.

**ErrUnknown** Es gibt keine Session mit dieser ID.

**ErrUnavailable** Die Session ist bereits einer anderen Prozedurinstanz  
zugeordnet.

**\_SvcSessionDelete** - Sitzung löschen

Die Anweisung `SvcSessionControl(_SvcSessionDelete)` löscht die Session-ID und die Zuordnung der Session zur laufenden Prozedur. Die Session-ID ist danach nicht mehr vorhanden. Nach dem Befehl kann aber eine neue Session mit `_SvcSessionCreate` oder eine vorhandene mit `_SvcSessionLoad` zugeordnet werden.

Folgende Fehlerresultate sind möglich:

**ErrSvcSessionState** Die Prozedur hat keine zugeordnete Session.

## Kontakt

### Verarbeitungshinweise zum SOA-Service

#### Hinweise zur Verarbeitung innerhalb des SOA-Service

Der SOA-Service kann für unterschiedliche Zwecke verwendet werden. Im Folgenden wird an Beispielen erläutert, wie Funktionen des SOA-Service für bestimmte Anwendungsfälle programmiert werden können.

#### Task mit Betriebsart TIME

Beim Einsatz des SOA-Service in der Betriebsart TIME ist zunächst zwischen kurz laufenden Ereignisprozeduren (im Bereich bis 10 Sekunden) und länger laufenden Verarbeitungen zu unterscheiden. Bei längeren Verarbeitungen muss die Möglichkeiten bestehen, den Task und somit die laufende Ereignisprozedur innerhalb von wenigen Sekunden zu beenden. Dies geschieht beispielsweise beim Beenden des Systemdienstes oder beim Herunterfahren des Rechners. Die Ereignisprozedur muss daher den Zustand der Eigenschaft StopRequest regelmäßig überprüfen und sich im Falle eines angeforderten Stopps auch beenden.

Es ist zu beachten, dass die gesetzte Ausführungszeit für das nächste Ereignis

(SvcCallDelay oder SvcCallTime) nicht garantiert ist. Bei einem Neustart des Tasks wird das nächste Ereignis unmittelbar nach dem Start des Task ausgelöst und damit meist früher als vorgesehen. Wird der Rechner beispielsweise zu Wartungszwecken komplett heruntergefahren, findet das nächste Ereignis erst nach dem erneuten Systemstart statt und daher oft später als vorgesehen. Zur Kontrolle dieser Zustände kann der Zeitpunkt des letzten Ereignisses durch die Prozedur in der Datenbank gespeichert werden.

#### Beispiel:

```
sub SvcTime1( aObjHdl          : handle;      // Task-Objekt   aEvtType        : int;
               // in einer Stunde oder zur nächsten vollen Stunde wieder aufrufen tCTS->vpMinutes # 0;  tCTS-
```

#### Task mit Betriebsart Socket

Die einfachste Verarbeitungsform eines SOCKET-Services besteht aus vier Schritten:

- Verbindungsauftbau durch die Gegenstelle
- Empfangen einer Anfrage der Gegenstelle
- Senden einer Antwort an die Gegenstelle
- Verbindungsabbau durch den Service

Für das Empfangen der Anfrage und das Senden der Antwort ist das verwendete Kommunikationsprotokoll entscheidend. Neben Standardprotokollen wie HTTP, SMTP, POP3 oder Telnet können auch eigene Protokolle verwendet werden. Textbasierte Protokolle wie Telnet haben den Vorteil, dass Anfragen und Antworten ganz oder teilweise in Textzeilen organisiert sind, die sehr einfach mit SckRead(..., SckLine, ...) gelesen beziehungsweise mit SckWrite(..., SckLine, ...) geschrieben werden können. Bei der Verwendung des HTTP kann das Objekt HTTP verwendet werden. Bei binären Protokollen muss dagegen mit vordefinierten oder datenbezogenen Bytelängen gearbeitet werden.

Die Verarbeitungsschritte 2 und 3 können sich je nach Protokoll wiederholen. Die gesamte Sitzung wird dabei in einem einzigen Ereignis durchgeführt. Dabei sollte der Socket-Timeout für SckRead() und SckWrite() nicht zu niedrig eingestellt sein, da es

## Kontakt

ansonsten zu dem Fehler ErrTimeout kommen kann, wenn die Gegenseite nicht schnell genug ist oder Kommunikationsverzögerungen eintreten.

Analog zu TIME-Services müssen Ereignisprozeduren, die längere Zeit laufen, den Zustand der Eigenschaft StopRequest regelmäßig überprüfen und sich im Falle eines angeforderten Stopps auch beenden.

Beispiel:

```
sub SvcSocket1( aObjHdl           : handle;           // Task-Objekt   aEvtType           : int;
```

Task mit Betriebsart SOCKET mit Keep-Alive Keep-Alives ermöglichen die Verwendung einer Socket-Verbindung über einen längeren Zeitraum, ohne das während der gesamten Verbindungsduer eine Prozedur läuft und somit ein Datenbankbenutzer benötigt wird. Bei Verwendung des HTTP-Protokolls kann die Steuerung von Keep-Alives am einfachsten durch die Verwendung von HTTP-Objekten erfolgen. Bei anderen Protokollen müssen die Ereignisprozeduren dies selbst steuern.

Die Keep-Alive-Zeit bestimmt, für welche maximale Zeitspanne die Socket-Verbindung nach dem Ende der Ereignisprozedur noch offen bleibt, um weitere Daten der Gegenstelle empfangen zu können. In der Konfigurationsdatei des Tasks kann mit der Einstellung socket keepalive ein Standardwert für diese Zeitspanne eingestellt werden, die Verwendung von Keep-Alives wird jedoch immer durch die Ereignisprozedur gesteuert.

Am Anfang einer Verbindung steht das Ereignis SckEvtConnect. In der Ereignisprozedur gibt es jetzt drei Möglichkeiten, die Verbindung zu behandeln:

Mit SckClose() wird die Verbindung getrennt und es folgen keine weiteren Ereignisse für die Verbindung.

Mit SckInfo(<handle>, SckKeepAlive, <time>) wird eine Keep-Alive-Zeit eingestellt. Die Verbindung bleibt bestehen und das nächste Ereignis ist entweder SckEvtData, SckEvtTimeout oder SckEvtDisconnect. Die Keep-Alive-Zeit gilt nur bis zum nächsten Ereignis, ein weiteres Keep-Alive muss erneut eingestellt werden. Die Keep-Alive-Zeitangabe erfolgt in Millisekunden, alternativ kann die Konstante SckDefaultKeepAlive verwendet werden, wodurch die für den Task voreingestellte Zeitspanne aus der Konfigurationsdatei verwendet wird.

Wenn weder die erste noch die zweite Variante benutzt wird, erfolgt als nächstes Ereignis SckEvtDisconnect für diese Verbindung.

Sofern ein Keep-Alive gesetzt wird und innerhalb der Zeit Daten von der Gegenstelle eintreffen, wird das Ereignis SckEvtData ausgelöst. In diesem Ereignis können dann weitere Daten empfangen und verarbeitet werden und gegebenenfalls auch Daten zurückgesendet werden. In der Ereignisprozedur gibt es dieselben oben genannten drei Möglichkeiten zur weiteren Steuerung der Verbindung.

Treffen innerhalb der Keep-Alive-Zeit keine Daten der Gegenstelle ein, erfolgt das Ereignis SckEvtTimeout. Die Verbindung ist nach wie vor offen, in diesem Ereignis könnten auch Daten an die Gegenstelle gesendet werden. Die Prozedur muss an dieser

## Kontakt

Stelle entscheiden, ob die Verbindung weiter bestehen bleibt. In diesem Fall muss ein erneutes Keep-Alive gesetzt werden. Andernfalls kann ein SckClose() erfolgen (kein weiteres Ereignis) oder die Prozedur beendet sich (SckEvtDisconnect erfolgt).

Das Ereignis SckEvtDisconnect wird ausgelöst, wenn in den Ereignissen

SckEvtConnect, SckEvtData oder SckEvtTimeout weder ein Keep-Alive gesetzt wird noch ein SckClose() erfolgt. Außerdem erfolgt der Aufruf, wenn innerhalb einer Keep-Alive-Zeit die Gegenstelle die Verbindung beendet.

Beispiel:

```
sub SvcWork( aSck          : handle;           // Socket Deskriptor): logic; local { tReq
sub SvcSocket2( aObjHdl    : handle;           // Task-Objekt aEvtType   : int;
```

### Task mit Betriebsart SOCKET mit Keep-Alive und HTTP

Um einen Socket-Service mit dem HTTP-Protokoll zu realisieren, ist die Verwendung des HTTP-Objekts die einfachste Möglichkeit. Dabei übernehmen die HTTP-Objekte für das Empfangen der Anfrage und das Senden der Antwort auch die Steuerung des Keep-Alives. Bei Verwendung von HTTP/1.0 wird generell ohne Keep-Alive gearbeitet. Bei HTTP/1.1 bestimmt die Anfrage der Gegenstelle und die Antwort des Socket-Service ob Keep-Alive verwendet wird. Enthält die Anfrage den HTTP-Kopfeintrag Connection: close wird kein Keep-Alive verwendet. Wird beim Senden der Antwort bei HttpClose( HttpCloseConnection) benutzt, wird ebenfalls kein Keep-Alive aktiviert.

Die Verbindung bleibt nach der Ereignisprozedur also nur unter folgenden Bedingungen bestehen:

Es wird für das Empfangen der Anfrage und das Senden der Antwort jeweils ein entsprechendes HTTP-Objekt verwendet.

Sowohl die Anfrage als auch die Antwort verwenden HTTP/1.1.

Die Anfrage enthält kein "Connection: close" im HTTP-Header.

Die Antwort wird ohne die Option HttpCloseConnection gesendet.

Beispiel (verkürzt):

```
sub SvcSocket3( aObjHdl    : handle;           // Task-Objekt aEvtType   : int;
local {  tSck        : handle;           // Socket Deskriptor  tRequest     : handle;
```

### Task mit Betriebsart SOCKET mit Sessions

Sowohl bei der Verwendung von Keep-Alives als auch bei mehreren Verbindungen von einer Gegenstelle aus besteht oft die Notwendigkeit, die Kommunikation als eine zusammenhängende Sitzung (Session) zu verarbeiten. Dazu muss bei mehreren aufeinanderfolgenden Anfragen die Gegenstelle identifiziert werden. Da nach jedem Aufruf einer Ereignisprozedur eventuelle globale Daten und Feldinhalte verlorengehen, können Sitzungsdaten entweder mit der Anweisung SvcSessionControl(), in der Datenbank oder als zentrale Datenobjekte abgelegt werden.

Sofern die Verbindung durch den Einsatz von Keep-Alives bestehen bleibt, kann die Gegenstelle zuverlässig anhand von Quell-IP-Adresse (SckInfo(..., SckAddrPeer)) und

## Kontakt

Port (SckInfo(..., SckPortPeer)) bestimmt werden.

Da bei mehreren separaten Verbindungen der Quell-Port wechselt, muss in diesen Fällen eine Identifikation über das Protokoll erfolgen. Bei HTTP ist dies zum Beispiel relativ einfach durch die Verwendung von Cookies möglich. In dem Cookie kann die Session-ID, die von der Anweisung SvcSessionControl() zurückgegeben wurde, übertragen werden. Es bestehen aber auch die Möglichkeiten die Id in der URI oder als Parameter des Requests zu übergeben. Die Kommunikation über das HTTP wird durch das Objekt HTTP unterstützt. In anderen Fällen ist je nach verwendetem Protokoll die Verarbeitung von Sitzungen aufwändiger oder gar nicht möglich.

Da Sitzungen ohne eine Information von der Gegenstelle abgebrochen werden können, muss es für die temporären Sitzungsdaten eine Funktion zum Aufräumen abgelaufener Daten geben, die beispielsweise über einen TIME-Service realisiert werden kann und die abgelaufene Daten anhand eines Zeitstempels identifizieren kann.

## Jobs

Der primäre Zweck von Jobs ist der Verarbeitung von Aufgaben in Zusammenhang mit einem Ereignis, aber außerhalb der Ereignisprozedur selbst. Beim TIME-Service kann dies der Fall sein, wenn beispielsweise die Ereignisprozedur in Sekundenabständen aufgerufen wird und sie lediglich die Verarbeitung von Aufgaben initiiert, sie aber nicht selbst durchführt. Beim SOCKET-Service dauert die Verarbeitung unter Umständen länger, wodurch die Gegenstelle eventuell einen Timeout-Fehler bekommt, bevor die Antwort gesendet werden kann. In diesem Fall könnten temporäre Antworten ("In Bearbeitung") gesendet werden, während die eigentliche Verarbeitung ein Job erledigt. Dies ist oft einfacher realisierbar als die Kommunikation und die Verarbeitung in nur einer Prozedur zu implementieren.

Der einfachste Fall bei Jobs ist "Start and forget it". Der Starter überprüft lediglich, ob der Start erfolgreich ist, das Resultat und der Verarbeitungsverlauf sind nicht von Interesse. Das oben genannte Beispiel für den TIME-Service könnte wie folgt aussehen.

### Beispiel:

```
sub JobWork( aObjHdl      : handle;           // Task-Objekt aEvtType    : int;
sub SvcTime2( aObjHdl      : handle;           // Task-Objekt aEvtType    : int;
```

Falls der Starter den Verarbeitungszustand und das Jobende erfahren will, muss mit JobOpen() ein Kontroll-Objekt angelegt werden, welches am Ende mit JobClose() auch wieder entfernt wird. Die Eigenschaft JobErrorCode ist 0 (ErrOk), solange der Job läuft und ErrTerminated, wenn der Job korrekt beendet wurde. Da in der Jobverarbeitung aber auch benutzerdefinierte Fehler entstehen können, kann dies über die Eigenschaft JobStatus signalisiert werden.

### Beispiel (Ausschnitt):

```
tJobCtrl # JobOpen(tJobID);if (tJobCtrl){ // Schleife läuft, solange der Job läuft while (tJobC
```

## Kontakt

In bestimmten Fällen wartet die Ereignisprozedur aber gar nicht das Ende der Jobverarbeitung ab, sondern beendet sich vorher. Die Gegenstelle sendet dann beispielsweise erneut eine Anfrage, um das Resultat des Jobs abzuholen. In diesem Fall kann die Ereignisprozedur erneut ein JobOpen() auf den Job durchführen, sofern ihr der benötigte Job-Deskriptor bekannt ist (diesen kann die Gegenstelle beispielsweise im Cookie mitsenden). Für ein solches Verfahren ist es notwendig, dass auch nach dem Ende der Jobprozedur der Job-Deskriptor gültig ist und ein JobOpen() funktioniert. Zu diesem Zweck wird beim Start des Jobs eine Zeitspanne angegeben, in der der Job-Deskriptor nach dem Ende der Jobprozedur noch für die Anweisung verwendet werden kann:

```
// Job starten mit 5 Minuten ID-Gültigkeit nach JobendetJobID # JobStart(_JobThread, 300, 'Service
```

Wenn beim Ende der Jobprozedur bereits ein Kontrollobjekt vorhanden ist, bleibt dies natürlich bis zum JobClose() vorhanden.

### Jobs mit Verarbeitungsschleifen

In vielen Anwendungen endet der Job nach erfolgreicher Erledigung der Aufgabe. Unter Umständen soll die Jobprozedur noch weitere Aufgaben übernehmen. Der Verarbeitung weiterer Aufgaben kann in Form einer Schleife realisiert werden, die entweder auf Daten der Kontrollprozedur wartet (siehe Kommunikation mit Jobs) oder in einen Wartezustand geht. Mit dem Befehl JobSleep() kann die Jobprozedur in einen Wartezustand versetzt werden, der im Gegensatz zu SysSleep() vor Ablauf der Zeit enden kann. Führt ein Kontrollobjekt mit der Funktion JobControl() eine der Optionen JobWakeUp, JobStop oder JobTerminate durch, wird JobSleep() abgebrochen und die Jobprozedur kann entsprechend reagieren.

### Beispiel (Ausschnitt aus der JobProzedur):

```
while (aObjHdl->spStopRequest = false){ // Verarbeitung ... // auf ein Ereignis warten while
```

### Kommunikation mit Jobs

Jobprozeduren können mit dem Kontroll-Objekt über MSX-Befehle bidirektional Daten austauschen. Dazu werden auf beiden Seiten jeweils zwei Kommunikationskanäle zum Senden und Empfangen angelegt. Findet die Kommunikation nur in eine Richtung statt, benötigt jede Seite auch nur einen Kanal.

Sendet eine der beiden Seite eine Nachricht, muss diese zunächst komplett sein, bevor sie für die andere Seite sichtbar wird. Es kann also noch nicht mit dem Lesen einer Nachricht begonnen werden, falls auf der anderen Seite noch kein MsxWrite( MsxEnd, 0) erfolgt ist. Die fertige Nachricht wird zunächst im Hauptspeicher vorgehalten und die Eigenschaft JobMsxReadQ auf der Gegenseite um eins erhöht. Anschließend kann die andere Seite die Nachricht öffnen und den Inhalt auslesen. Nach dem Öffnen der Nachricht reduziert sich JobMsxReadQ wieder um eins.

Auf der Seite des Senders wird bei der Ausführung als Thread die Eigenschaft JobMsxWriteQ um eins erhöht, solange die Nachricht noch nicht von der anderen Seite geöffnet wurde. Bei der Ausführung als Prozess werden geschriebene Nachrichten immer sofort zum Zielprozess übertragen und dort zwischengespeichert. Daher ist die Eigenschaft JobMsxWriteQ bei der Verwendung von Jobprozessen immer

## Kontakt

0.

Sind beim Beenden der Jobprozedur noch Nachrichten vorhanden, können diese alle noch von der Kontrollprozedur gelesen werden. Umgekehrt bleiben beim Beenden der Kontrollprozedur noch alle Nachrichten für die Jobprozedur erhalten. Erst beim Schließen des Kommunikationskanals auf beiden Seiten werden ungelesene Nachrichten gelöscht.

Für MsxRead() kann jede Seite einen eigenen Timeout über den Befehl JobControl(..., JobMsxTimeoutRead, ...) setzen. Höhere Timeout-Werte stellen kein Problem dar, da beim Stoppen des Tasks oder dem Stoppen des Jobs die Funktionen MsxRead() oder MsxWrite() abgebrochen werden und der Fehler ErrTerminated zurückgeliefert wird. Die gilt allerdings nicht für MsxRead()- oder MsxWrite()-Operationen auf Sockets oder Dateien.

Beispiel:

```
sub JobWork( aObjHdl           : handle;      // Task-Objekt   aEvtType          : int;
             // Nachricht empfangen    if (tError = _rOk)      {           tMsxHdlR->MsxRead(_MsxItem, tMsgItem);
             { ... try {     // Job starten    tJobID   # JobStart(_JobThread, 0, 'Service:JobWork');    tJo
```

Jobmodus (Threads vs. Prozesse)

Jobs können entweder als eigener Thread im laufenden Taskprozess oder als separater Betriebssystemprozess gestartet werden.

Im Thread-Modus kann die Job-Prozedur sehr schnell gestartet werden, da beispielsweise die Datenstruktur im Speicher kopiert werden kann und nicht erneut aus der Datenbank geladen werden muss. Die Kommunikation mit dem Kontroll-Objekt kann direkt und ohne nennenswerten Overhead erfolgen. Der zusätzliche Thread benötigt nur geringe Ressourcen und wird garantiert zusammen mit dem Taskprozess beendet.

Im Prozess-Modus läuft hingegen der Job unabhängig vom Taskprozess. Ein Absturz des Jobprozesses führt somit nicht zum Beenden des Taskprozesses. Dies kann bei Verarbeitungen mit DLLs oder mit COM (Component Object Model) von Vorteil sein. Außerdem verfügt der Jobprozess über einen vollständig eigenen Adressraum, der nicht mit anderen Jobs oder dem Taskprozess geteilt werden muss. Der Start eines Jobs im Prozess-Modus dauert etwas länger als im Thread-Modus und der Job verbraucht auch mehr Ressourcen des Betriebssystems. Beispielsweise sind für die Kommunikation mit dem Taskprozess zwei zusätzliche Threads erforderlich.

## Kontakt

### Job



Job-Objekt des SOA-Service, Standard- oder Advanced-Clients SOA-Service,  
Siehe Eigenschaften,  
JobStart()

Der Job wird im Rahmen einer Ereignisfunktion eines Tasks, dem Standard- oder Advanced-Client mit der Anweisung JobStart() erzeugt. Der Deskriptor auf das Job-Objekt wird beim Aufruf der zugehörenden Ereignisfunktion übergeben.

In den Eigenschaften des Objekts sind unter anderem Informationen zu dem aufrufenden Task, Statusinformationen und den verwendeten Ressourcen abgelegt.

## Kontakt

### JobControl



**Objekt zur Job-Kontrolle**  
SOA-Service,

Siehe Eigenschaften,  
JobOpen()

Nachdem ein Job im Standard-, Advanced-Clients oder in einer Ereignisfunktion eines Tasks gestartet wurde, kann ebenfalls ein JobControl-Objekt zur Steuerung und Kontrolle des Jobs erzeugt werden. Dieses Objekt wird mit JobOpen() angelegt und ist vom Typ HdlJobControl mit dem Untertyp JobThread oder JobProcess.

## Kontakt

### Task

 **Task-Objekt des SOA-Service**  
SOA-Service,

Siehe Eigenschaften,  
Job

Das Task-Objekt wird vom SOA-Service erzeugt, sobald dieser gestartet wird. Für jeden Task steht ein entsprechenden Objekt bereit. Der Deskriptor des entsprechenden Objekts wird an die Ereignisfunktion übergeben.

Die Eigenschaften des Objekts beinhalten unter anderem Informationen über den Task und die verwendeten Ressourcen.

## Kontakt

### Der CONZEPT 16-Laufwerkstreiber

#### Beschreibung des Laufwerkstreibers

Der CONZEPT 16-Laufwerkstreiber stellt eine Schnittstelle zum Dateisystem zur Verfügung. Beim Start des Laufwerkstreibers wird ein Laufwerksbuchstabe oder eine Netzwerkfreigabe (oder beides) angegeben. Alle Aktionen des Betriebssystems oder von anderen Programmen auf diesem Laufwerk werden vom Laufwerkstreiber erfasst und durch Prozeduren innerhalb der Datenbank durchgeführt. Das Laufwerk verhält sich dabei, wie ein gewöhnliches Laufwerk.

Mit dem Laufwerkstreiber können Inhalte aus der Datenbank in Form von Verzeichnissen und Dateien aufbereitet und dem Benutzer oder anderen Programmen zur Verfügung gestellt werden.

Die Installation wird im Abschnitt Installation des Laufwerkstreibers beschrieben. Die generelle Funktionsweise befindet sich in Arbeitsweise des Laufwerkstreibers. Die Beschreibung, wie der Laufwerkstreiber eingesetzt werden kann, befindet sich im Abschnitt Referenzimplementation. Die Referenzimplementation ist auch in der Beispieldatenbank "CodeLibrary" zu finden.

## Kontakt

### Installation des Laufwerkstreibers

#### Beschreibung der Installation des Laufwerkstreibers

Die Dateien des Laufwerkstreibers werden über die CONZEPT 16 Installationsroutine eingerichtet. Die notwendigen Dateien werden in die Verzeichnisse System32 und System32\drivers des Windows-Systemverzeichnisses kopiert.

System32\cbfsconnectevtmsg.dll

System32\cbfsconnectMntNtf2017.dll

System32\cbfsconnectNetRdr2017.dll

System32\drivers\cbfsconnect2017.sys



Der Laufwerkstreiber, der verarbeitende Client und der Datenbank-Server müssen aus Performanzgründen auf dem gleichen System gestartet werden.

Die Konfiguration des Laufwerkstreibers erfolgt über Konfigurationsdateien. Die Einstellungen in den Konfigurationsdateien sind im Abschnitt SOA-Service - Konfigurationsdatei beschrieben. Ein Beispiel befindet sich im Abschnitt Referenzimplementation - Vorbereitung der Anwendung.

## Kontakt

### Arbeitsweise des Laufwerkstreibers

#### Beschreibung der Arbeitsweise des Laufwerkstreibers

Der Laufwerkstreiber stellt ein Laufwerk zur Verfügung, dessen Inhalt durch die Funktionen des entsprechenden CONZEPT 16-Clients zur Verfügung gestellt werden.

Alle Aktionen auf dem Laufwerk lösen ein entsprechendes Ereignis beim Laufwerkstreiber aus. Dieser baut eine Verbindung über einen definierten Socket (siehe SOA-Service - Konfigurationsdatei) auf, sendet eine Anfrage (Request) in Form eines Datenpakets und wartet, bis er eine Antwort (Response) bekommt. Anschließend kann die Verbindung wieder getrennt werden. Da häufig innerhalb kurzer Zeit erneut Ereignisse auf dem Laufwerk ausgelöst werden, empfiehlt es sich für die Verbindung ein Keep-alive von 60 Sekunden zu definieren. Dadurch wird die Performanz gegenüber Verbindungen ohne Keep-alive erheblich gesteigert.

Da die Kommunikation über einen Socket stattfindet, können sowohl der SOA-Service (SOCKET-basiert), als auch der Standard- oder Advanced-Client (Ereignis EvtSocket) zur Ausführung der entsprechenden Funktionen verwendet werden. Diese Funktionen stellen die angeforderten Informationen zusammen und senden sie in Form von Datenpaketen zurück an den Laufwerkstreiber. Die Informationen werden dem entsprechenden Programm zur Verfügung gestellt.



Für den normalen Betrieb muss der SOA-Service verwendet werden, da sonst der Laufwerkstreiber erst nach dem Starten der Applikation gestartet werden kann und das Laufwerk nur bis zum Beenden der Applikation zur Verfügung steht.

Nach dem Starten des Laufwerkstreibers wird sofort das Ereignis DrvInit durchgeführt. Damit das Ereignis verarbeitet werden kann, muss der CONZEPT 16-SOA-Service bzw. der Client vor dem Laufwerkstreiber gestartet werden. Wird der Laufwerkstreiber angehalten, wird das Ereignis DrvTerm durchgeführt. Sollten zu diesem Zeitpunkt noch Dateien auf dem Laufwerk geöffnet sein, werden Änderungen an diesen Dateien nicht übernommen.

Kommt es zu einem Verarbeitungsfehler innerhalb des Laufwerkstreibers, wird das Ereignis DrvError ausgelöst. In diesem Ereignis kann eine Protokollierung vorgenommen werden. Alle Ereignisse werden in dem Bereich Ereignisse des Laufwerkstreibers beschrieben.

Der Laufwerkstreiber verfügt über einen eigenen Cache für die Verzeichnisstruktur, die Dateien, die Stammdaten des Laufwerks und die Benutzerinformationen (einschließlich Berechtigungen). Die Aktualität der Informationen wird über entsprechende Timeouts gesteuert. Die Timeouts werden getrennt voneinander in den Response-Datenpaketen von DrvGetFileEntries, DrvGetVolumeData und DrvLoginDomainUser angegeben. Informationen, die innerhalb des Timeouts angefordert werden, führen nicht dazu, dass ein Request gesendet wird. Die Informationen werden aus dem Cache geladen. Für Dateien wird bei jedem Zugriff DrvOpenFile aufgerufen. Der Wert im Item ItemContentChanged bestimmt, ob DrvReadFile ausgelöst wird (true) oder die Datei aus dem Cache geladen wird (false).

Zu jedem Ereignis sendet der Laufwerkstreiber ein Datenpaket und wartet auf eine Antwort. Die Datenpakete sind bei den Ereignissen beschrieben. Der generelle Aufbau

## Kontakt

der Datenpakete kann dem Abschnitt Beschreibung der Datenpakete entnommen werden.



Der Laufwerkstreiber, der verarbeitende Client und der Datenbank-Server müssen aus Performanzgründen auf dem gleichen System gestartet werden.

Nach dem Versenden der Antwort kann die Socket-Verbindung wieder getrennt werden. Wird hier allerdings die Verbindung mit einem Keep-Alive offen gehalten, kann eine erheblich höhere Performanz erreicht werden. Empfehlenswert ist ein Keep-Alive von 60 Sekunden.

Es können mehrere Laufwerke erzeugt werden. Dazu müssen beim SOA-Service mehrere Laufwerkstreiber und die dazugehörigen SOCKET-Tasks definiert werden. Über das Task-Objekt können die verschiedenen Laufwerke unterschieden werden.

Treten bei der Verarbeitung durch den Laufwerkstreiber Fehlerzustände auf, werden diese in der Protokolldatei des Laufwerkstreibers eingetragen (siehe Log-Einträge).

## Kontakt

### Beschreibung der Datenpakete

#### Beschreibung zum Aufbau der Datenpakete zwischen dem Laufwerkstreiber und der Applikation

Die Kommunikation mit dem Laufwerkstreiber erfolgt über Datenpakete, die mit Hilfe der Msx-Befehle gelesen bzw. erstellt werden können. Erfolgt eine Aktion auf dem Laufwerk wird vom Laufwerkstreiber ein Datenpaket gesendet. Das Datenpaket hat folgenden Aufbau:

```
<MessageID> <ItemHeader> <ProtocolID> <ProtocolVersion>...
```

Die **<MessageID>** ist eine Nummer, die das Ereignis auf dem Laufwerk identifiziert (siehe Ereignisse des Laufwerkstreibers). Die folgenden Werte sind fest vorgegeben (**<ItemHeader>** = 1000, **<ProtocolID>** = 0xF3A49E52, **<ProtocolVersion>** = 0x00040000). Mit diesen Werten kann festgestellt werden, ob die korrekte Version des Laufwerkstreibers installiert ist. Der Kopf des Datenpaket kann über folgende Anweisungen gelesen werden:

```
tDrvMsxRead # MsxOpen(_MsxSocket | _MsxRead, tSck); tDrvMsxRead->MsxRead(_MsxMessage, tRequestMess
```

Abhängig vom Ereignis folgen anschließend weitere Informationen. Diese Informationen werden immer durch ein Msx-Item (vom Datentyp int) eingeleitet. Abhängig von dem Msx-Item können dann die Informationen gelesen werden. Welche Informationen zur Verfügung stehen, ist in den entsprechenden Ereignissen beschrieben. Für das Auslesen der Items und der dazugehörigen Informationen, bietet sich eine Schleife an, die solange liest, bis das komplette Datenpaket ausgewertet wurde. Eine komplette Liste der Informationsbereiche befindet sich im Abschnitt DrvItem...

```
tDrvMsxRead # MsxOpen(_MsxSocket | _MsxRead, tSck); tDrvMsxRead->MsxRead(_MsxMessage, tRequestMess
```

Der Laufwerkstreiber wartet immer auf eine Antwort auf seine Nachricht. Die Nachricht enthält immer eine Message-Id, Informationen über das Protokoll und einen Resultatwert. Weitere Inhalte hängen von dem entsprechenden Ereignis ab und sind dort beschrieben.

```
<MessageID> <ItemHeader> <ProtocolID> <ProtocolVersion>... <ItemResult> <Result>
```

Im Beispiel eine Antwort auf das Initialisierungsereignis nach dem Starten des Laufwerkstreibers:

```
tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSck); tDrvMsxWrite->MsxWrite(_MsxMessage, (tRequest
```

Die Reihenfolge, in der die einzelnen Bereiche (Msx-Items) in der Antwort definiert werden, spielt keine Rolle. Alle Zeichenketten innerhalb der Datenpakete sind im UTF-8-Format und müssen entsprechend gewandelt werden.

## Kontakt

### \_DrvItem...

Beschreibung der Informationsbereich der Datenpakete

#### Beschreibung

Siehe der

#### Datenpakete

Die Datenpakete (Request und Response) sind in Informationsbereiche unterteilt. Jeder Informationsbereich wird mit einem Msx-Item eingeleitet. Dieses Item hat eine Nummer, die mit den hier beschriebenen Konstanten verglichen werden kann. Der Wert der Konstanten steht in Klammern dahinter. In einem Informationsbereich sind eine fest vorgegebene Anzahl von Informationen hinterlegt. Bei den Ereignissen sind die im Request und Responds vorhandenen Informationsbereiche beschrieben. Im folgenden befindet sich eine Liste alle Bereiche.

### \_DrvItemAuthUserDomain (1050)

Informationsbereich des am Betriebssystem angemeldeten Benutzers.

alpha Benutzername des angemeldeten Benutzers alpha

Domäne des angemeldeten Benutzers

### \_DrvItemAuthAccessId (1051) Informationsbereich der

Zugriffs-Berechtigungen.

int Zugriffs-Id

### \_DrvItemCacheTime (1015)

Informationsbereich für den Zeitpunkt, zu dem die Datei oder das Verzeichnis in den Cache geschrieben wurde. Der Zeitpunkt wird als Zeitstempel angegeben. Die Verarbeitung kann in einer Variable von Typ caltime erfolgen. Der Wert muss dann mit CnvCB() gewandelt werden.

bigint Datum und Uhrzeit der letzten Abfrage

### \_DrvItemCfgCacheDisk (1022)

Informationsbereich über den Cache des Laufwerkstreiber. Standardmäßig wird das Verzeichnis **DRIVE\_CACHE** im temporären Pfad des Betriebssystems angelegt. Wird ein anderer Pfad angegeben, wird dort ebenfalls das Verzeichnis **DRIVE\_CACHE** erzeugt. Das Verzeichnis wird beim Beenden des Laufwerkstreibers entfernt.



Der Inhalt des Verzeichnisses wird beim Start vollständig geleert. Sollten dort Dateien (z. B. anderer Programme) enthalten sein, sind diese nach dem Start des Laufwerkstreibers nicht mehr vorhanden. Es ist daher wichtig, dass dieses Verzeichnis exklusiv dem Laufwerkstreiber zur Verfügung steht.

bigint maximale Größe des Caches

alpha Pfad für temporäre Dateien

### \_DrvItemCfgIgnoreMaskRequest (1024)

## Kontakt

Informationsbereich der Dateimaske. In diesem Bereich werden Dateimasken angegeben, für die keine DrvGetFileInfo-Nachrichten erzeugt werden sollen.

alpha Pipe-separierte () Liste

\_DrvItemCfgMountLocal (1020) Informationsbereich

über das lokale Laufwerk.

alpha Laufwerksbuchstabe

\_DrvItemCfgMountNetwork (1021) Informationsbereich

über die Netzwerkfreigabe.

alpha Freigabename

\_DrvItemComputerName (1013)

In diesem Informationsbereich wird der Name des Computers übergeben, auf dem der Laufwerkstreiber gestartet wurde.

alpha Name des Computers

\_DrvItemContentChanged (1012)

Informationsbereich für geänderte Daten. Der Laufwerkstreiber hat einen eigenen Cache. Haben sich die Daten von einer Datei nicht geändert, kann in diesem Abschnitt angegeben werden, dass die Daten nicht erneut übertragen werden müssen.

logic Daten geändert

\_DrvItemDriverData (1014)

Informationen über den eingesetzten Laufwerkstreiber.

alpha Versionsnummer des installierten Treibers

\_DrvItemErrorText (1010)

Informationsbereich für Fehlertexte.

alpha Fehlertext

\_DrvItem FileAccess (1062)

Informationsbereich für den Zugriffsmodus auf eine Datei.

int Zugriffs-Flags

\_DrvItemFileAttributes (1071)

Informationsbereich für die Dateiattribute.

int Dateiattribute

\_DrvItemFileAuthorisation (1064)

## Kontakt

Informationsbereich für Zugriffsrechte.

**int Zugriffs-Id**

**int Berechtigungen**

Die Berechtigung kann als Kombination der folgenden Konstanten angegeben werden:

<b>_DrvAuthAll</b>	<b>Alle Rechte vorhanden</b>
<b>_DrvAuthNone</b>	<b>Keine Rechte vorhanden</b>
<b>_DrvAuthList</b>	<b>Dateien und Verzeichnisse listen</b>
<b>_DrvAuthDelete</b>	<b>Dateien und Verzeichnisse löschen</b>
<b>_DrvAuthRename</b>	<b>Dateien und Verzeichnisse umbenennen</b>
<b>_DrvAuthMove</b>	<b>Dateien und Verzeichnisse verschieben</b>
<b>_DrvAuthSetAttributes</b>	<b>Attribute von Dateien und Verzeichnissen setzen</b>
<b>_DrvAuthRead</b>	<b>Datei lesen</b>
<b>_DrvAuthWrite</b>	<b>Datei schreiben</b>
<b>_DrvAuthExecute</b>	<b>Datei ausführen</b>
<b>_DrvAuth.CreateDirectory</b>	<b>Verzeichnis erstellen</b>
<b>_DrvAuthCreateFile</b>	<b>Datei erstellen</b>
<b>_DrvItemFileContentDisk (1065)</b>	

Informationsbereich für den Inhalt der Datei.

**alpha Pfad- und Dateiname der Datei im temporären Pfad**

**\_DrvItemFileCustom (1063) Informationsbereich für**

**benutzerdefinierte Daten**

**alpha benutzerdefinierte Informationen**

**\_DrvItemFileHash (1067)**

Informationsbereich für die Prüfsumme der Datei. Die Prüfsumme kann vom Programmierer verwendet werden, um eine Änderung an einer Datei festzustellen. Es kann dann entschieden werden, ob die Dateiinformationen erneut übertragen werden müssen.

**alpha Prüfsumme**

**\_DrvItemFileName (1061) Informationsbereich**

**für den Dateinamen.**

**alpha Name der Datei**

**\_DrvItemFileNameNew (1069) Informationsbereich für**

**den neuen Dateinamen. alpha neuer Dateiname**

## Kontakt

**\_DrvItemFilePath (1060) Informationsbereich für den Pfad der Datei.**

**alpha Pfad (ohne Laufwerksbuchstaben)**

**\_DrvItemFilePathNew (1068)**

**Informationsbereich für den neuen Pfad. Der Informationsbereich wird nur beim Verschieben einer Datei benötigt.**

**alpha neuer Dateipfad**

**\_DrvItemFileSize (1070) Informationsbereich für die**

**Größe der Datei.**

**bigint Größe der Datei in Bytes**

**\_DrvItemFileSubData (1073)**

**Informationsbereich für untergeordnete Daten. Dieser Bereich ist nur für Verzeichniseinträge relevant. Sind in dem Unterverzeichnis weder Dateien noch weitere Unterverzeichnisse, kann das hier angegeben werden. Ein Auswerten dieses Verzeichnisbaumes ist dann nicht notwendig.**

**logic Verzeichnis leer**

**\_DrvItemFileTime (1072)**

**Informationsbereich für Datum und Uhrzeiten der Datei. Datum und Uhrzeit werden als Zeitstempel dargestellt. Die Verarbeitung kann in einer Variable von Typ caltime erfolgen. Der Wert muss dann mit CnvCB() bzw. CnvBC() gewandelt werden.**

**bigint Datum und Uhrzeit der Dateierzeugung bigint**

**Datum und Uhrzeit des letzten Zugriffs bigint Datum  
und Uhrzeit der letzten Änderung**

**\_DrvItemResult (1001)**

**Resultat der Operation. In diesem Abschnitt des Request-Datenpakets wird der Fehlerwert des Laufwerkstreibers angegeben. Bei einem Response-Datenpaket kann in diesem Abschnitt ein Resultat angegeben werden, das entweder an die Applikation weitergeleitet wird, die die Aktion auf dem Laufwerk ausgelöst hat, oder vom Laufwerkstreiber selbst ausgewertet wird.**

**int Resultat-Wert**

**\_DrvItemTempPath (1011)**

**Informationsbereich für die Speicherung im temporären Pfad.**

**alpha Pfad- und Dateiname im temporären Pfad**

**\_DrvItemTimeout (1002)**

## Kontakt

Informationsbereich für das zeitliche Verhalten der Applikation. Hier werden Angaben gemacht, wann eine Aktion noch mal durchgeführt werden soll.

**int Zeitraum in Millisekunden**

**\_DrvItemVolumeId (1041)** Informationsbereich für die Laufwerksnummer.

**int Laufwerks-Id**

**\_DrvItemVolumeLabel (1042)** Informationsbereich für den Laufwerknamen.

**alpha Name des Laufwerks**

**\_DrvItemVolumeSize (1040)**

Informationsbereich für die Laufwerksgröße und des freien Speicherplatzes.

**bignum freier Speicherbereich in Bytes**

**bignum gesamte Speicherkapazität in Bytes**

## Kontakt

### Ereignisse des Laufwerkstreibers

#### Der Laufwerkstreiber unterstützt folgende Ereignisse

Der Laufwerkstreiber erstellt zu den im folgenden angegebenen Ereignissen auf dem Laufwerk ein Datenpaket. Anschließend wartet er auf eine Antwort. Das Antwort-Datenpaket muss in den entsprechenden Funktionen des SOA-Tasks erstellt werden. Nach dem Austausch der Nachrichten kann die Socket-Verbindung wieder getrennt werden, es können aber erheblich mehr Anfragen pro Sekunde verarbeitet werden, wenn die Verbindung mit einem Keep-alive für mindestens 60 Sekunden offen gehalten wird. Das Keep-alive muss bei jedem Versenden einer Antwort gesetzt werden:

```
tSck->SckInfo(_SckKeepAlive, 60000); // Keep-alive 60 sec
```

[DrvCloseFile](#)  
[DrvCreateFile](#)  
[DrvDeleteFile](#)  
[DrvError](#)  
[DrvFlushFile](#)  
[DrvGetFileEntries](#)  
[DrvGetFileInfo](#)  
[DrvGetVolumeData](#)  
[DrvInit](#)  
[DrvLoginDomainUser](#)  
[DrvMoveFile](#)  
[DrvOpenFile](#)  
[DrvReadFile](#)  
[DrvRenameFile](#)  
[DrvTerm](#)

## Kontakt

### **DrvInit**

Aufruf beim Start des Laufwerkstreibers

#### **Request**

<b>MessageId</b>	<u>_DrvReqInit</u>	<b>Id des Ereignisses</b>
<b>ItemHeader</b>	<u>_DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
<b>ProtocolId</b>	<b>0xF3A49E52</b>	<b>Id des verwendeten Protokolls</b>
<b>ProtocolVersion</b>	<b>0x00040000</b>	<b>Version des verwendeten Protokolls</b>
<b>ItemComputerName</b>	<u><b>DrvItemComputerName</b></u>	<b>Informationsbereich des Computers</b>
<b>ComputerName</b>	<u><b>alpha(80)</b></u>	<b>Name des Rechners auf dem der Laufwerkstreiber gestartet ist</b>
<b>ItemDriverData</b>	<u><b>DrvItemDriverData</b></u>	<b>Informationsbereich des Treibers</b>
<b>DriverData</b>	<u><b>alpha(80)</b></u>	<b>Versionsnummer des installierten Treibers</b>

#### Siehe

#### **Verwandte Befehle**

Dieses Ereignis wird beim Start des Laufwerkstreibers aufgerufen. In dem übergebenen Datenpaket wird der Name des Rechners, auf dem der Laufwerkstreiber gestartet wurde (ComputerName), und die Version des Treibers (DriverVersion) übergeben.

Der Treiber erwartet folgendes Datenpaket:

### **Response**

<b>MessageId</b>	<u><b>_DrvResInit</b></u>	<b>Id der Antwort</b>
<b>ItemHeader</b>	<u><b>_DrvItemHeader</b></u>	<b>Kopf des Datenpakets</b>
<b>ProtocolId</b>	<b>0xF3A49E52</b>	<b>Id des verwendeten Protokolls</b>
<b>ProtocolVersion</b>	<b>0x00040000</b>	<b>Version des verwendeten Protokolls</b>
<b>ItemCfgMountLocal</b>	<u><b>DrvItemCfgMountLocal</b></u>	<b>Informationsbereich des lokalen Laufwerks</b>
<b>DriveLetter</b>	<u><b>alpha(1)</b></u>	<b>Der Laufwerksbuchstabe des lokalen Laufwerks</b>
<b>ItemCfgMountNetwork</b>	<u><b>DrvItemCfgMountNetwork</b></u>	<b>Informationsbereich für Netzwerkfreigaben</b>
<b>ShareName</b>	<u><b>alpha(80)</b></u>	<b>Freigabename des Laufwerks</b>
<b>ItemCfgCacheDisk</b>	<u><b>DrvItemCfgCacheDisk</b></u>	<b>Informationsbereich für den Cache</b>

## Kontakt

**CacheMaxSize**

bigint

Maximaler Festplattenspeicher, der zur Zwischenspeicherung der Inhalte verwendet wird. Es wird ein entsprechendes Verzeichnis im temporären Pfad des Betriebssystems angelegt.

**CachePath**

alpha(1024)

Laufwerk und Pfad für temporäre Dateien

Informationsbereich für Dateifilter

Hier kann eine Pipe-separierte Liste mit Dateimasken angegeben werden.

Für die entsprechenden Dateien werden keine Dateiinformationen (siehe [DrvGetFileInfo](#)) abgefragt.

Ergebnis-Bereich

Ergebniswert

**ItemResult**

DrvItemResult

**Result**

int

Damit ein Laufwerk erstellt werden kann, muss mindestens einer der Bereiche ItemCfgMountLocal oder ItemCfgMountNetwork angegeben werden. Wird nur eine Netzwerkfreigabe erzeugt, kann das Laufwerk auch noch zu einem späteren Zeitpunkt einem Laufwerksbuchstaben zugewiesen werden. Sind beide Bereiche angegeben, wird eine Netzwerkfreigabe erzeugt und der angegebene Laufwerksbuchstabe zugewiesen.

Ein freier Laufwerksbuchstabe kann mit der Anweisung [FsiDiskInfo\(..., FsiDiskExists\)](#) ermittelt werden.

Im Bereich ItemCfgCacheDisk wird die maximale Speichernutzung und der Pfad für temporäre Dateien angegeben. In dem Verzeichnis wird das Unterverzeichnis DRIVE\_CACHE angelegt. Wird kein Pfad angegeben, erfolgt die Speicherung in dem Unterverzeichnis DRIVE\_CACHE im Windows-Pfad für temporäre Dateien. Das Verzeichnis wird beim Beenden des Laufwerkstreibers entfernt.



Sind zu diesem Zeitpunkt noch veränderte Dateien geöffnet, kann das Verzeichnis nicht entfernt werden (siehe [DrvTerm](#)).



Der Inhalt des Verzeichnisses wird beim Start vollständig geleert. Sollten dort Dateien (z. B. anderer Programme) enthalten sein, sind diese nach dem Start des Laufwerkstreibers nicht mehr vorhanden. Es ist daher wichtig, dass dieses Verzeichnis exklusiv dem Laufwerkstreiber zur Verfügung steht.

Die Angabe des Bereichs ItemCfgIgnoreMaskRequest ist optional.

## Kontakt

Wird in Result ErrOk angegeben, ist die Initialisierung erfolgreich. Bei ErrDrvQuit wird der Laufwerkstreiber beendet. Wird ein anderer Fehlerwert angegeben, startet sich der Laufwerkstreiber nach einer Minute erneut.

Kann das angegebene Laufwerk oder die Netzwerkf freigabe nicht erzeugt werden, wird ein Fehler generiert und das Ereignis DrvError aufgerufen.

Beispiel:

```
if (tDrvRequestMessageId = _DrvReqInit){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSck);
```

Werden mehrere Laufwerke eingerichtet, muss jedes Laufwerk einen eigenen Laufwerksbuchstaben und/oder Freigabenamen bekommen. Jedes Laufwerk benötigt einen eigene SOA-Task. Die Anfragen von den verschiedenen Laufwerken können dann über die Namen der Tasks unterschieden werden.

```
switch (aSvcHdl->spSvcName){ case 'DrvSoaSocketPicture' : { DrvSend:DrvCfgMountLocal('T');
```

## Kontakt

### DrvTerm

Aufruf beim Beenden des Laufwerkstreiber

Request

MessageId	<u>_DrvReqTerm</u>	Id des Ereignisses
ItemHeader	<u>_DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
ProtocolId	0xF3A49E52	<b>Id des verwendeten Protokolls</b>
ProtocolVersion	0x00040000	<b>Version des verwendeten Protokolls</b>

Siehe [Verwandte Befehle](#)

Dieses Ereignis wird aufgerufen, wenn der Laufwerkstreiber beendet wird. Hier können Ressourcen, die in der Initialisierung (siehe [DrvInit](#)) reserviert wurden, wieder freigegeben werden.

### Response

MessageId	<u>_DrvResTerm</u>	<b>Id der Antwort</b>
ItemHeader	<u>_DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
ProtocolId	0xF3A49E52	<b>Id des verwendeten Protokolls</b>
ProtocolVersion	0x00040000	<b>Version des verwendeten Protokolls</b>
ItemResult	<u>_DrvItemResult</u>	<b>Ergebnis-Bereich</b>
Result	<u>int</u>	<b>Ergebniswert</b>

Die Angabe eines Resultats ist optional.

Sind zu diesem Zeitpunkt noch geänderte Dateien geöffnet, kann das temporäre Verzeichnis (siehe [DrvInit](#)) nicht entfernt werden, da diese Dateien in dem Verzeichnis verbleiben. Die Dateien können zum Wiederherstellen von Informationen genutzt werden.

Beispiel:

```
if (tDrvRequestMessageId = _DrvReqTerm) {    tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSck);
```

## Kontakt

### DrvError

Aufruf beim Auftreten eines Fehlers

Request

MessageId	<u>_DrvReqError</u>	Id des Ereignisses
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemResult	<u><a href="#">_DrvItemResult</a></u>	Informationsbereich für das Resultat
Result	<u>int</u>	Fehlerwert
ItemErrorText	<u><a href="#">_DrvItemErrorText</a></u>	Informationsbereich des Fehler-Textes
ErrorText	<u>alpha(250)</u>	Fehler-Text
Siehe	<u>Verwandte Befehle</u>	

Tritt bei der Verwendung des Laufwerkstreibers ein Verarbeitungsfehler auf, wird dieses Ereignis aufgerufen. In dem Ereignis kann eine entsprechende Protokollierung vorgenommen werden. Der Wert in Result kann mit folgenden Konstanten verglichen werden:

[ErrDrvDriverNotInstalled](#)

Der Grätetreiber des Laufwerkstreibers ist nicht installiert.

[ErrDrvInitFailed](#)

Fehler bei der Initialisierung des Laufwerkstreibers.

[ErrDrvInitInvalidData](#)

Kein Laufwerksbuchstabe und Freigabename angegeben.

[ErrDrvInitDriveLetterInUse](#) Der

Laufwerksbuchstabe ist in Benutzung.

[ErrDrvCreateTempPath](#)

Temporärer Pfad ist ungültig oder konnte nicht angelegt werden.

[ErrDrvCreateTempFile](#)

Temporäre Datei konnte nicht angelegt werden.

[ErrDrvClearTempPath](#)

Temporärer Pfad war nicht leer und wurde geleert.

Response

MessageId	<u>_DrvResError</u>	Id der Antwort
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

Beispiel:

```
if (tRequestMessageId = _DrvReqError) { // read error message tDrvMsxRead->MsxRead(_MsxItem, tIt
```

## Kontakt

<b>DrvLoginDomainUser</b>	
Aufruf beim Anmelden eines Benutzers	
<b>Request</b>	
<b>MessageId</b>	<u>_DrvReqLoginDomainUser</u> Id des Ereignisses
	Kopf des
<b>ItemHeader</b>	<u>_DrvItemHeader</u>
	Datenpakets
<b>ProtocolId</b>	<b>0xF3A49E52</b>
	Id des verwendeten
	Protokolls
<b>ProtocolVersion</b>	<b>0x00040000</b>
	Version des
	verwendeten
	Protokolls
	Informationsbereich
	des am
<b>ItemAuthUserDomain</b>	<u>_DrvItemAuthUserDomain</u> Betriebssystem
	angemeldeten
	Benutzers
<b>UserName</b>	<u>alpha(80)</u>
	Benutzername des
	angemeldeten
<b>DomainName</b>	<u>alpha(80)</u>
	Benutzers
	Domain des
	angemeldeten
<b>ItemAuthAccessId</b>	<u>_DrvItemAuthAccessId</u>
	Informationsbereich
	des Benutzers
<b>AccessId</b>	<u>int</u>
	Zugriffs-Id des
	Benutzers
<b>Siehe</b>	<u>Verwandte Befehle</u>
Dieses Ereignis wird bei der Anmeldung eines Benutzers ausgelöst und wenn eine erneute Abfrage des Benutzers erfolgt. Erfolgt die erneut Abfrage aufgrund des zuvor angegebenen Timeouts, wird die bisherige AccessId im Datenpaket des Requests mit übergeben.	
<b>Response</b>	
<b>MessageId</b>	<u>_DrvResLoginDomainUser</u> Id der Antwort
<b>ItemHeader</b>	<u>_DrvItemHeader</u>
	Kopf des Datenpakets
<b>ProtocolId</b>	<b>0xF3A49E52</b>
	Id des verwendeten Protokolls
<b>ProtocolVersion</b>	<b>0x00040000</b>
	Version des verwendeten Protokolls
<b>ItemAuthAccessId</b>	<u>_DrvItemAuthAccessId</u>
	Informationsbereich des Benutzers
<b>AccessId</b>	<u>int</u>
	Zugriffs-Id des Benutzers
<b>ItemTimeout</b>	<u>_DrvItemTimeout</u>
	Informationsbereich für den Timeout
<b>Timeout</b>	<u>int</u>
	Zeit bis zur erneuten Abfrage des
	Benutzers
<b>ItemResult</b>	<u>_DrvItemResult</u>
	Ergebnis-Bereich
<b>Result</b>	<u>int</u>
	Ergebniswert
Wird in Result ein Wert ungleich 0 ( <u>ErrOk</u> ) angegeben, wird der Benutzer abgewiesen.	
Er kann dann auf dem Laufwerk keine Aktionen durchführen.	

## Kontakt

Die übergebene Zugriffs-Id bestimmt die Berechtigungen des Benutzers. Die Berechtigungen werden für jede Zugriffs-Id bei den entsprechenden Ereignissen (zum Beispiel DrvGetFileEntries) im Informationsbereich der Berechtigungen (ItemFileAuthorisation) angegeben.

### Beispiel:

Sollen zwei unterschiedliche Gruppen von Benutzern zugreifen, müssen zwei Zugriff-Ids definiert werden (zum Beispiel 10 und 12). Für die Zugriffs-Id 10 werden alle Berechtigungen gesetzt, während für die Zugriffs-Id 12 nur lesend auf die Daten zugegriffen werden kann. In diesem Ereignis wird ein Benutzer des Betriebssystems zu einer Zugriffs-Id zugewiesen und hat somit entweder alle Berechtigungen oder nur lesenden Zugriff.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqLoginDomainUser){  tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrit
```

## Kontakt

### DrvGetVolumeData

Aufruf bei der Anforderung der Laufwerksdaten  
Request

MessageId	<u>_DrvReqGetVolumeData</u>	Ereignisse
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	<b>0xF3A49E52</b>	Id des verwendeten Protokolls
ProtocolVersion	<b>0x00040000</b>	Version des verwendeten Protokolls

Siehe [Verwandte Befehle](#)

Dieses Ereignis wird aufgerufen, wenn die Informationen des Laufwerks abgefragt werden (zum Beispiel durch den Aufruf der Eigenschaften eines Laufwerks). Als Ergebnis wird die gesamte und die freie Speicherkapazität sowie die Id und der Name des Laufwerks zurückgegeben.

### Response

MessageId	<u>_DrvResGetVolumeData</u>	Id der Antwort
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	<b>0xF3A49E52</b>	Id des verwendeten Protokolls
ProtocolVersion	<b>0x00040000</b>	Version des verwendeten Protokolls
ItemVolumeSize	<u><a href="#">DrvItemVolumeSize</a></u>	Informationsbereich für den Speicherplatz
FreeSize	<u><a href="#">bigint</a></u>	Freier Speicherplatz auf dem Laufwerk (in Bytes)
TotalSize	<u><a href="#">bigint</a></u>	Gesamtkapazität des Laufwerks (in Bytes)
ItemVolumeId	<u><a href="#">DrvItemVolumeId</a></u>	Informationsbereich für die Id des Laufwerks
VolumeId	<u><a href="#">int</a></u>	Id des Laufwerks
ItemVolumeLabel	<u><a href="#">DrvItemVolumeLabel</a></u>	Informationsbereich des Laufwerknamens.
VolumeLabel	<u><a href="#">alpha(80)</a></u>	Name des Laufwerks.
ItemTimeout	<u><a href="#">DrvItemTimeout</a></u>	Informationsbereich für den Timeout
Timeout	<u><a href="#">int</a></u>	Zeit bis zum erneuten Abruf der Informationen (in Millisekunden).

Beispiel:

```
if (tDrvRequestMessageId = _DrvReqGetVolumeData) {    tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite,
```

## Kontakt

### DrvGetFileEntries

Aufruf beim Lesen eines Ordners

#### Request

MessageId	<u>DrvReqGetFileEntries</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemAuthUserDomain	<u>DrvItemAuthUserDomain</u>	Informationsbereich Benutzerauthentifizierung
UserName	<u>alpha(80)</u>	Name des Benutzers
DomainName	<u>alpha(80)</u>	Name der Domain
ItemCacheTime	<u>DrvItemCacheTime</u>	Informationsbereich für die Cache-Nutzung
CacheTime	<u>bignum</u>	0 oder der Zeitpunkt, zu dem der Eintrag zuletzt abgefragt wurde
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich des Verzeichnisses
FilePath	<u>alpha(1024)</u>	Pfad des Verzeichnisses (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für den Programmierer
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen

#### Siehe

#### Verwandte Befehle

Das Ereignis wird aufgerufen, wenn ein Verzeichnis des Laufwerks gelesen wird. Die Informationen in dem Datenpaket beziehen sich auf das zu lesende Verzeichnis. Im Bereich ItemAuthUserDomain wird der am Betriebssystem angemeldete Benutzer übergeben.

Befindet sich der Eintrag bereits im Cache, wird in CacheTime der Zeitpunkt angegeben, zu dem der Eintrag zuletzt mit DrvGetFileEntries abgefragt wurde. Befindet sich der Eintrag nicht im Cache, wird 0 übergeben. Der Wert kann von dem in bignum gewandelten jetzigen Zeitpunkt abgezogen werden, um das Alter des Cache-Eintrags zu ermitteln.

```
tCaltimeNow->vmSystemTime(); tNow # CnvBC(tCaltimeNow); tInCacheTime # (tNow - tCacheTime) / 10000\
```

In FilePath und Custom01 befinden sich das zu lesende Verzeichnis und die benutzerdefinierten Informationen zu dem Verzeichnis. Wird das Wurzelverzeichnis abgefragt wird in FilePath '\' übergeben.

Der Laufwerkstreiber erwartet zunächst eine Nachricht mit Informationen, ob sich der Inhalt des Verzeichnisses geändert hat und einem Timeout. Für jeden Verzeichniseintrag wird eine weitere Nachricht gesendet. Das Timeout bestimmt, wann die Verzeichniseinträge erneut ermittelt werden sollen.

## Kontakt

<b>Response</b>		
<b>MessageId</b>	<u>DrvResGetFileEntries</u>	<b>Id der Antwort</b>
<b>ItemHeader</b>	<u>DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
<b>ProtocolId</b>	<b>0xF3A49E52</b>	<b>Id des verwendeten Protokolls</b>
<b>ProtocolVersion</b>	<b>0x00040000</b>	<b>Version des verwendeten Protokolls</b>
<b>ItemContentChanged</b>	<u>DrvItemContentChanged</u>	<b>Informationsbereich für Änderungen</b>
<b>ContentChanged</b>	<u>logic</u>	<b>Inhalt wurde geändert (true)</b>
<b>ItemTimeout</b>	<u>DrvItemTimeout</u>	<b>Informationsbereich für den Timeout</b>
<b>Timeout</b>	<u>int</u>	<b>Maximale Zeit bis zur nächsten Abfrage des Verzeichnisinhaltes (in Millisekunden)</b>
<b>ItemResult</b>	<u>DrvItemResult</u>	<b>Informationsbereich für das Ergebnis</b>
<b>Timeout</b>	<u>int</u>	<b>Bei einem Resultat ungleich 0 (<u>ErrOk</u>) werden keine Verzeichniseinträge übertragen</b>

**Wird in diesen Datenpaket ContentChanged = false angegeben, werden alle Informationen aus dem Cache gelesen. Hat sich der Inhalt des Verzeichnisses geändert, müssen alle Verzeichniseinträge einschließlich aller Berechtigungen übertragen werden.**

**Erfolgt eine weitere Abfrage des Verzeichnisinhaltes, bevor der Timeout abgelaufen ist, wird kein Ereignis ausgelöst, sondern der Inhalt aus dem Cache erneut zur Verfügung gestellt.**

**Für jeden Eintrag in dem Verzeichnis wird eine eigene Nachricht versendet:**

<b>Response</b>		
<b>MessageId</b>	<u>DrvResGetFileEntryData</u>	<b>Id der Antwort</b>
<b>ItemResult</b>	<u>DrvItemResult</u>	<b>Informationsbereich des Ergebnisses</b>
<b>Result</b>	<u>int</u>	<b>Ergebnis der Operation</b>
<b>ItemFileName</b>	<u>- DrvItemFileName</u>	<b>Informationsbereich für Dateinamen</b>
<b>FileName</b>	<u>alpha(1024)</u>	<b>Dateiname</b>
<b>ItemFileCustom</b>	<u>DrvItemFileCustom</u>	<b>Informationsbereich für den Programmierer</b>
<b>Custom01</b>	<u>alpha(4096)</u>	<b>Benutzerdefinierte Informationen</b>
<b>ItemFileSize</b>	<u>DrvItemFileSize</u>	<b>Informationsbereich für die Dateigröße</b>
<b>FileSize</b>	<u>bigint</u>	<b>Größe der Datei (in Bytes)</b>
<b>ItemFileAttributes</b>	<u>DrvItemFileAttributes</u>	<b>Informationsbereich für die Dateiattribute</b>
<b>FileAttributes</b>	<u>int</u>	<b>Dateiattribute</b>
<b>ItemFileTime</b>	<u>DrvItemFileTime</u>	<b>Informationsbereich für Dateidatum und -uhrzeit</b>
<b>CreationTime</b>	<u>bigint</u>	

## Kontakt

LastAccessTime	<u>bigint</u>	Datum und Uhrzeit der Dateierstellung
LastWriteTime	<u>bigint</u>	Datum und Uhrzeit des letzten Dateizugriffs
ItemFileSubData	<u>DrvItemFileSubData</u>	Datum und Uhrzeit der letzten Änderung
Empty	<u>logic</u>	Informationsbereich für weitere Daten
ItemFileAuthorisation	<u>DrvItemFileAuthorisation</u>	Die Datei ist leer ( <u>true</u> )
UserId	<u>int</u>	Informationsbereich für die Dateiberechtigungen Zugriffs-Id
Authorisation	<u>int</u>	Berechtigungen an der Datei (siehe Text)

Das Datenpaket für einen Verzeichniseintrag beinhaltet folgende Informationen:

### ItemFileName

In FileName wird der Dateiname oder der Name des Unterverzeichnisses angegeben, der angezeigt werden soll.

### ItemFileCustom

Der Bereich ist optional. In Custom01 können benutzerspezifische Informationen zu der Datei angegeben werden. Die Informationen werden nur dann geändert, wenn dieser Bereich im Antwortdatenpaket enthalten ist. Zum Löschen der benutzerdefinierten Daten muss ein Leerstring angegeben werden.

### ItemFileSize

FileSize beinhaltet die Größe der Datei (in Bytes). Bei der Rückgabe von Unterverzeichnissen darf dieser Bereich nicht angegeben werden.

### ItemFileAttributes

In FileAttributes werden die Attribute der Datei oder des Unterverzeichnisses angegeben.

Der Wert kann aus folgenden Konstanten kombiniert werden:

#### FsiAttrHidden Versteckte Datei

#### FsiAttrSystem Systemdatei

#### FsiAttrDir Verzeichnis

#### FsiAttrArchive Archivdatei

### ItemFileTime

In diesem Bereich wird das Datum und die Uhrzeit der Dateierstellung (CreationTime), des letzten Zugriffs (LastAccessTime) und der letzten Änderung (LastWriteTime) angegeben. Die Werte werden als Zeitstempel übergeben. Die Berechnung des Zeitstempels kann wie folgt durchgeführt werden:

```
tSystemCalTime->vmSystemTime() ; tDrvMsxWrite->MsxWrite(_MsxData, CnvBC(tSystemCalTime)) ;
```

Bei der Umwandlung wird die Zeitzone berücksichtigt.

## Kontakt

### ItemFileSubData

In Empty wird angegeben, ob ein Unterverzeichnis leer ist, d. h. ob in dem Verzeichnis Dateien und/oder weitere Unterverzeichnisse vorhanden sind. Wird true angegeben, ist das Verzeichnis leer. Der Bereich darf für Dateieinträge nicht angegeben werden.

### ItemFileAuthorisation

Der Bereich ItemFileAuthorisation muss für jede Zugriffs-Id angegeben werden. Können sich Benutzer mit zwei unterschiedlichen Zugriffs-Ids anmelden (siehe DrvLoginDomainUser), müssen hier zwei Bereiche mit den entsprechenden Berechtigungen angegeben werden. Die Berechtigungen werden als ODER-Kombination folgender Konstanten übergeben:

<u>DrvAuthAll</u>	Alle Rechte vorhanden
<u>DrvAuthNone</u>	Keine Rechte vorhanden
<u>DrvAuthList</u>	Dateien und Verzeichnisse listen
<u>DrvAuthDelete</u>	Dateien und Verzeichnisse löschen
<u>DrvAuthRename</u>	Dateien und Verzeichnisse umbenennen
<u>DrvAuthMove</u>	Dateien und Verzeichnisse verschieben
<u>DrvAuthSetAttributes</u>	Attribute von Dateien und Verzeichnissen setzen
<u>DrvAuthRead</u>	Datei lesen
<u>DrvAuthWrite</u>	Datei schreiben
<u>DrvAuthExecute</u>	Datei ausführen
<u>DrvAuth.CreateDirectory</u>	Verzeichnis erstellen
<u>DrvAuthCreateFile</u>	Datei erstellen

### ItemResult

Der Wert Result muss auf ErrOk gesetzt werden. Beim letzten Eintrag wird hier rNoRec angegeben.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqGetFileEntries){  tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileAttributes); // set attributes  tDrvMsxWrite->MsxW
Message for last entry  tDrvMsxWrite->MsxWrite(_MsxMessage, _DrvResGetFileEntryData);  tDrvM
tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileSubData); // set sub data//  tDrvMsxWrite->MsxWr
```

## Kontakt

### DrvGetFileInfo

Aufruf beim Abfragen von Dateiinformationen

#### Request

MessageId	<u>DrvReqGetFileInfo</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

**ItemAuthUserDomain** DrvItemAuthUserDomain Betriebssystem  
angemeldeten Benutzers

UserName	<u>alpha(80)</u>	Name des angemeldeten Benutzers
DomainName	<u>alpha(80)</u>	Domäne des angemeldeten Benutzers
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich der Datei Pfad- und Dateiname (ohne Laufwerksbuchstabe)
FilePath	<u>alpha(1024)</u>	Informationsbereich für benutzerdefinierte Daten
ItemFileCustom	<u>DrvItemFileCustom</u>	Benutzerdefinierte Informationen
Custom01	<u>alpha(4096)</u>	

#### Siehe

#### Verwandte Befehle

Dieses Ereignis wird aufgerufen, wenn Informationen zu einer Datei angefragt werden oder die Informationen von dem Ereignis DrvGetFileEntries zu alt sind. Für das Wurzelverzeichnis des Laufwerks wird dieses Ereignis aufgerufen, um die Berechtigungen für das Anlegen von Ordnern und Dateien zu ermitteln. Der beim Betriebssystem angemeldete Benutzer (ItemAuthUserDomain) und der Name der angeforderten Datei bzw. des Ordners (FilePath) werden in dem Datenpaket angegeben. Wird das Wurzelverzeichnis abgefragt wird hier '\' übergeben.

#### Response

MessageId	<u>DrvResGetFileInfo</u>	Id der Antwort
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für den Programmierer
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen

## Kontakt

<b>ItemFileSize</b>	<u><a href="#">DrvItemFileSize</a></u>	Informationsbereich für die Dateigröße
<b>FileSize</b>	<u><a href="#">bigint</a></u>	Größe der Datei (in Bytes)
<b>ItemFileAttributes</b>	<u><a href="#">DrvItemFileAttributes</a></u>	Informationsbereich für die Dateiattributte
<b>FileAttributes</b>	<u><a href="#">int</a></u>	Dateiattributte
<b>ItemFileTime</b>	<u><a href="#">DrvItemFileTime</a></u>	Informationsbereich für Dateidatum und -uhrzeit
<b>CreationTime</b>	<u><a href="#">bigint</a></u>	Datum und Uhrzeit der Dateierstellung
<b>LastAccessTime</b>	<u><a href="#">bigint</a></u>	Datum und Uhrzeit des letzten Dateizugriffs
<b>LastWriteTime</b>	<u><a href="#">bigint</a></u>	Datum und Uhrzeit der letzten Ändserung
<b>ItemFileSubData</b>	<u><a href="#">DrvItemFileSubData</a></u>	Informationsbereich für weitere Daten
<b>Empty</b>	<u><a href="#">logic</a></u>	Das Verzeichnis ist leer (true).
<b>ItemFileAuthorisation</b>	<u><a href="#">DrvItemFileAuthorisation</a></u>	Informationsbereich für die Dateiberechtigungen
<b>AccessId</b>	<u><a href="#">int</a></u>	Zugriffs-Id
<b>Authorisation</b>	<u><a href="#">int</a></u>	Berechtigungen an der Datei (siehe Text)
<b>ItemResult</b>	<u><a href="#">DrvItemResult</a></u>	Ergebnis-Bereich
<b>Result</b>	<u><a href="#">int</a></u>	Ergebniswert

In dem Datenpaket werden alle Informationen über eine Datei zurückgegeben. In dem Bereich ItemFileCustom können benutzerdefinierte Informationen angegeben werden. Diese werden bei allen Dateioperationen wieder mit übergeben. Die Angabe von ItemFileCustom ist optional. Wird der Wert nicht übergeben, bleiben die Daten unverändert. Sollen die Informationen gelöscht werden, muss eine leere Zeichenkette übergeben werden.

Die Größe der Datei wird in FileSize in Bytes angegeben. Bei den Dateiattributen (FileAttributes) können folgende Konstanten kombiniert werden:

[FsiAttrHidden](#) Versteckte Datei

[FsiAttrSystem](#) Systemdatei

[FsiAttrDir](#) Verzeichnis

[FsiAttrArchive](#) Archivdatei

In dem Bereich ItemFileTime wird das Datum und die Uhrzeit der Dateierzeugung, des letzten Zugriffs und der letzten Änderung in Form von Zeitstempeln angegeben. Der Zeitstempel kann wie folgt berechnet werden:

```
tSystemCalTime->vmSystemTime(); tDrvMsxWrite->MsxWrite(_MsxData, CnvBC(tSystemCalTime));
```

In diesem Fall wird die Systemzeit zur Berechnung des Zeitstempels verwendet. Bei der Umrechnung wird die Zeitzone berücksichtigt.

In Empty wird angegeben, ob das Unterverzeichnis leer ist (true). Der Abschnitt darf für Dateien nicht angegeben werden.

## Kontakt

Der Bereich ItemFileAuthorisation muss für jede Zugriffs-Id angegeben werden. Können sich Benutzer mit zwei unterschiedlichen Zugriffs-Ids anmelden (siehe DrvLoginDomainUser), müssen hier zwei Bereiche mit den entsprechenden Berechtigungen angegeben werden. Die Berechtigungen werden als OR-Kombination folgender Konstanten übergeben:

<u>DrvAuthAll</u>	Alle Rechte vorhanden
<u>DrvAuthNone</u>	Keine Rechte vorhanden
<u>DrvAuthList</u>	Dateien und Verzeichnisse listen
<u>DrvAuthDelete</u>	Dateien und Verzeichnisse löschen
<u>DrvAuthRename</u>	Dateien und Verzeichnisse umbenennen
<u>DrvAuthMove</u>	Dateien und Verzeichnisse verschieben
<u>DrvAuthSetAttributes</u>	Attribute von Dateien und Verzeichnissen setzen
<u>DrvAuthRead</u>	Datei lesen
<u>DrvAuthWrite</u>	Datei schreiben
<u>DrvAuthExecute</u>	Datei ausführen
<u>DrvAuth.CreateDirectory</u>	Verzeichnis erstellen
<u>DrvAuthCreateFile</u>	Datei erstellen

Wird in Result ErrFsiNoFile angegeben, wird der Fehler "Datei nicht vorhanden" ausgegeben. Die Dateiinformationen werden nur übertragen, wenn im Resultat ErrOk (0) angegeben wird.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqGetFileInfo){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, t
// tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileSubData); // set sub data// tDrvMsxWrite->MsxWr
```

## Kontakt

### DrvCreateFile

Aufruf beim Erzeugen einer Datei

#### Request

MessageId	<u>DrvReqCreateFile</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

#### ItemAuthUserDomain DrvItemAuthUserDomain Betriebssystem angemeldeten Benutzers

UserName	<u>alpha(80)</u>	Name des angemeldeten Benutzers
DomainName	<u>alpha(80)</u>	Domäne des angemeldeten Benutzers
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich für die Datei Pfad und Dateiname (ohne Laufwerksbuchstabe)
FilePath	<u>alpha(1024)</u>	Informationsbereich für die Zugriffsart Zugriffsart (siehe Text)
ItemFileAccess	<u>DrvItemFileAccess</u>	Informationsbereich für Dateiattribute Dateiattribute
FsiFlags	<u>int</u>	
ItemFileAttributes	<u>DrvItemFileAttributes</u>	
FileAttributes	<u>int</u>	
Siehe	<u>Verwandte Befehle</u>	

Dieses Ereignis wird aufgerufen, wenn eine Datei oder ein Verzeichnis auf dem Laufwerk erzeugt wird. In dem Request wird der am Betriebssystem angemeldete Benutzer übergeben. In FilePath befindet sich der Pfad (ohne Laufwerksbuchstaben) und der Dateiname der neu angelegten Datei. Der Inhalt von FsiFlags kann mit folgenden Konstanten, oder Kombinationen davon, verglichen werden:

<u>FsiAcsR</u>	nur Lesezugriff
<u>FsiAcsW</u>	nur Schreibzugriff
<u>FsiAcsRW</u>	Lese- und Schreibzugriff
<u>FsiDenyNone</u>	kein exklusiver Zugriff
<u>FsiStdRead</u>	Standard-Lesemodus
<u>FsiStdWrite</u>	Standard-Schreibmodus

Die Dateiattribute stehen in FileAttributes und können mit folgenden Konstanten verglichen werden:

## Kontakt

FsiAttrHidden Versteckte Datei

FsiAttrSystem Systemdatei

FsiAttrDir Verzeichnis

FsiAttrArchive Archivdatei

**Response**

<u>MessageId</u>	<u>DrvResCreateFile</u>	<b>Id der Antwort</b>
<u>ItemHeader</u>	<u>DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
<u>ProtocolId</u>	<u>0xF3A49E52</u>	<b>Id des verwendeten Protokolls</b>
<u>ProtocolVersion</u>	<u>0x00040000</u>	<b>Version des verwendeten Protokolls</b>
<u>ItemFileCustom</u>	<u>DrvItemFileCustom</u>	<b>Informationsbereich für benutzerdefinierte Daten</b>
<u>Custom01</u>	<u>alpha(4096)</u>	<b>Benutzerdefinierte Informationen</b>
<hr/>		
<u>ItemFileAuthorisation</u> <u>DrvItemFileAuthorisation</u>		
<u>UserId</u>	<u>int</u>	<b>Informationsbereich für Berechtigungen</b>
<u>Authorisation</u>	<u>int</u>	<b>Zugriffs-Id</b>
<u>ItemResult</u>	<u>DrvItemResult</u>	<b>Berechtigungen</b>
<u>Result</u>	<u>int</u>	<b>Ergebnis-Bereich</b>
		<b>Ergebniswert</b>

Der Bereich ItemFileCustom ist optional. Die übergebenen Informationen in Custom01 werden bei jeder Dateioperation wieder übertragen. Hier können vom Programmierer Informationen zur späteren Verarbeitung angegeben werden.

Der Bereich ItemFileAuthorisation muss für jede Zugriffs-Id angegeben werden.

Können sich Benutzer mit zwei unterschiedlichen Zugriffs-Ids anmelden (siehe DrvLoginDomainUser), müssen hier zwei Bereiche mit den entsprechenden Berechtigungen angegeben werden. Die Berechtigungen werden als ODER-Kombination folgender Konstanten übergeben:

<u>DrvAuthAll</u>	<b>Alle Rechte vorhanden</b>
<u>DrvAuthNone</u>	<b>Keine Rechte vorhanden</b>
<u>DrvAuthList</u>	<b>Dateien und Verzeichnisse listen</b>
<u>DrvAuthDelete</u>	<b>Dateien und Verzeichnisse löschen</b>
<u>DrvAuthRename</u>	<b>Dateien und Verzeichnisse umbenennen</b>
<u>DrvAuthMove</u>	<b>Dateien und Verzeichnisse verschieben</b>
<u>DrvAuthSetAttributes</u>	<b>Attribute von Dateien und Verzeichnissen setzen</b>
<u>DrvAuthRead</u>	<b>Datei lesen</b>
<u>DrvAuthWrite</u>	<b>Datei schreiben</b>
<u>DrvAuthExecute</u>	<b>Datei ausführen</b>
<u>DrvAuth.CreateDirectory</u>	<b>Verzeichnis erstellen</b>
<u>DrvAuthCreateFile</u>	<b>Datei erstellen</b>

Die Datei wird nur dann erzeugt, wenn in Result ErrOk (0) angegeben wird. Durch das Setzen eines ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

**Beispiel:**

```
if (tDrvRequestMessageId = _DrvReqCreateFile){  tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, ts
```

## Kontakt

### DrvDeleteFile

Aufruf beim Löschen einer Datei

#### Request

MessageId	<u>DrvReqDeleteFile</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

**ItemAuthUserDomain** DrvItemAuthUserDomain Betriebssystem  
angemeldeten Benutzers

UserName alpha(80) Name des angemeldeten Benutzers

DomainName alpha(80) Domäne des angemeldeten Benutzers

ItemFilePath DrvItemFilePath Informationsbereich der Datei

FilePath alpha(1024) Path- und Dateiname (ohne Laufwerksbuchstabe)

ItemFileCustom DrvItemFileCustom Informationsbereich für benutzerdefinierte Daten

Custom01 alpha(4096) Benutzerdefinierte Informationen

#### Siehe Verwandte Befehle

Dieses Ereignis wird aufgerufen, wenn eine Datei gelöscht wird. Die zu löschen Datei wird in FilePath angegeben.

#### Response

MessageId	<u>DrvResDeleteFile</u>	Id der Antwort
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemResult	<u>DrvItemResult</u>	Ergebnis-Bereich
Result	int	Ergebniswert

Die Datei wird nur dann gelöscht, wenn in Result ErrOk (0) angegeben wird. Durch das Setzen eines ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

Beispiel:

```
if (tDrvRequestMessageId = _DrvReqDeleteFile){  tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, ts
```

## Kontakt

### DrvMoveFile

Aufruf beim Verschieben einer Datei

#### Request

MessageId	<u>DrvReqMoveFile</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

**ItemAuthUserDomain** DrvItemAuthUserDomain Betriebssystem  
angemeldeten Benutzers

UserName	<u>alpha(80)</u>	Name des angemeldeten Benutzers
DomainName	<u>alpha(80)</u>	Domäne des angemeldeten Benutzers
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha(1024)</u>	Pfad- und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen
ItemFilePathNew	<u>DrvItemFilePathNew</u>	Informationsbereich des neuen Pfades
FilePathNew	<u>alpha(1024)</u>	Neuer Pfad der Datei
Siehe	<u>Verwandte Befehle</u>	

Das Ereignis wird ausgelöst, wenn eine Datei verschoben wird. Der neue Pfadname befindet sich in FilePathNew.

#### Response

MessageId	<u>DrvResMoveFile</u>	Id der Antwort
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten (optional)
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen
ItemFileSize	<u>DrvItemFileSize</u>	Informationsbereich der Dateigröße

## Kontakt

FileSize	<u>bigint</u>	(optional) Größe der Datei (in Bytes)
ItemFileAttributes	<u>DrvItemFileAttributes</u>	Informationsbereich für Dateiattribute (optional)
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit (optional)
CreationTime	<u>bigint</u>	Zeitstempel der Erzeugung der Datei
LastAccessTime	<u>bigint</u>	Zeitstempel des letzten Zugriffs
LastWriteTime	<u>bigint</u>	Zeitstempel der letzten Änderung
ItemFileAuthorisation	<u>DrvItemFileAuthorisation</u>	Informationsbereich für Berechtigungen (optional)
UserId	<u>int</u>	Zugriffs-ID
Authorisation	<u>int</u>	Berechtigungen
ItemResult	<u>DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</u>	Ergebniswert

Die Bereiche ItemFileCustom, ItemFileSize, ItemFileAttributes, ItemFileTime und ItemFileAuthorisation sind optional. Die Informationen werden nur dann geändert, wenn diese Bereiche im Antwortdatenpaket enthalten ist. Zum Löschen der Daten muss ein Leerstring oder 0 (für die int-Werte) angegeben werden. Bei ItemFileTime muss immer ein gültiger Zeitstempel angegeben werden, wenn dieser gesetzt wird.

Die Datei wird nur dann verschoben, wenn in Result ErrOk (0) angegeben wird. Durch das Setzen eines ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqMoveFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSck
```

## Kontakt

### DrvRenameFile

Aufruf beim Umbenennen einer Datei

#### Request

MessageId	<u>DrvReqRenameFile</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

**ItemAuthUserDomain** DrvItemAuthUserDomain Betriebssystem  
angemeldeten Benutzers

UserName	<u>alpha(80)</u>	Name des angemeldeten Benutzers
DomainName	<u>alpha(80)</u>	Domäne des angemeldeten Benutzers
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha(1024)</u>	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen
ItemFileNameNew	<u>DrvItemFileNameNew</u>	Informationsbereich des neuen Dateinamens
FileNameNew	<u>alpha(1024)</u>	Neuer Name der Datei

#### Siehe Verwandte Befehle

Das Ereignis wird ausgelöst, wenn eine Datei umbenannt wird. Der neue Dateiname befindet sich in FileNameNew.

#### Response

MessageId	<u>DrvResRenameFile</u>	Id der Antwort
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten (optional)
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen

## Kontakt

<b>ItemFileSize</b>	<u><a href="#">DrvItemFileSize</a></u>	Informationsbereich der Dateigröße (optional)
<b>FileSize</b>	<u><a href="#">bigint</a></u>	Größe der Datei (in Bytes)
<b>ItemFileAttributes</b>	<u><a href="#">DrvItemFileAttributes</a></u>	Informationsbereich für Dateiattribute (optional)
<b>FileAttributes</b>	<u><a href="#">int</a></u>	Dateiattribute
<b>ItemFileTime</b>	<u><a href="#">DrvItemFileTime</a></u>	Informationsbereich für Dateidatum und -uhrzeit (optional)
<b>CreationTime</b>	<u><a href="#">bigint</a></u>	Zeitstempel der Erzeugung der Datei
<b>LastAccessTime</b>	<u><a href="#">bigint</a></u>	Zeitstempel des letzten Zugriffs
<b>LastWriteTime</b>	<u><a href="#">bigint</a></u>	Zeitstempel der letzten Änderung
<b>ItemFileAuthorisation</b>	<u><a href="#">DrvItemFileAuthorisation</a></u>	Informationsbereich für Berechtigungen (optional)
<b>UserId</b>	<u><a href="#">int</a></u>	Zugriffs-Id
<b>Authorisation</b>	<u><a href="#">int</a></u>	Berechtigungen
<b>ItemResult</b>	<u><a href="#">DrvItemResult</a></u>	Ergebnis-Bereich
<b>Result</b>	<u><a href="#">int</a></u>	Ergebniswert

Die Bereiche ItemFileCustom, ItemFileSize, ItemFileAttributes, ItemFileTime und ItemFileAuthorisation sind optional. Die Informationen werden nur dann geändert, wenn diese Bereiche im Antwortdatenpaket enthalten ist. Zum Löschen der Daten muss ein Leerstring oder 0 (für die int-Werte) angegeben werden. Bei ItemFileTime muss immer ein gültiger Zeitstempel angegeben werden, wenn dieser gesetzt wird.

Die Datei wird nur dann umbenannt, wenn in Result [ErrOk](#) (0) angegeben wird. Durch das Setzen eines [ErrFsi...](#)-Wertes können andere Fehlerzustände übermittelt werden.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqRenameFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, ts
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## Kontakt

### DrvReadFile

Aufruf beim Lesen einer Datei

#### Request

MessageId	<u>DrvReqReadFile</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

**ItemAuthUserDomain** DrvItemAuthUserDomain Betriebssystem angemeldeten Benutzers

UserName	<u>alpha(80)</u>	Name des angemeldeten Benutzers
DomainName	<u>alpha(80)</u>	Domäne des angemeldeten Benutzers
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha(1024)</u>	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha(4096)</u>	Benutzerdefinierte Informationen
ItemFileHash	<u>DrvItemFileHash</u>	Informationsbereich der Prüfsumme
FileHash	<u>alpha(250)</u>	Prüfsumme der Datei
ItemCacheTime	<u>DrvItemCacheTime</u>	Informationsbereich für die Cache-Nutzung
CacheTime	<u>bint</u>	0 oder der Zeitpunkt, zu dem die Datei zuletzt abgefragt wurde
ItemTempPath	<u>DrvItemTempPath</u>	Informationsbereich für den temporären Pfad
TempPath	<u>alpha(1024)</u>	Pfad und Dateiname im temporären Pfad

Siehe Verwandte Befehle

Dieses Ereignis wird ausgelöst, wenn eine Datei gelesen wird. In dem Bereich ItemAuthUserDomain wird der Benutzer und die Domäne des beim Betriebssystem

## Kontakt

angemeldeten Benutzers übergeben. In FilePath und Custom01 befinden sich der Pfad und Dateiname auf dem Laufwerk und die benutzerdefinierten Informationen der Datei. In FileHash wird die Prüfsumme der Datei übertragen, die beim letzten Aufruf von DrvReadFile gesetzt wurde. Befindet sich die Datei bereits im Cache, wird in CacheTime der Zeitpunkt angegeben, zu dem die Datei zuletzt mit DrvReadFile abgefragt wurde. Befindet sich die Datei nicht im Cache, wird 0 übergeben. Der Inhalt der Datei muss in das temporäre Verzeichnis ausgelagert werden. Das Laufwerk und der Pfad dazu befindet sich in TempPath.

Über die Prüfsumme kann der Programmierer entscheiden, ob sich die Datei in der Zwischenzeit geändert hat und neu übertragen werden muss.

Das Antwort-Datenpaket darf erst versendet werden, wenn die Informationen vollständig zur Verfügung stehen.

Der Laufwerkstreiber erwartet folgendes Datenpaket als Antwort:

### Response

MessageId	<u>DrvResReadFile</u>	<b>Id der Antwort</b>
ItemHeader	<u>DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
ProtocolId	<b>0xF3A49E52</b>	<b>Id des verwendeten Protokolls</b>
ProtocolVersion	<b>0x00040000</b>	<b>Version des verwendeten Protokolls</b>
ItemContentChanged	<u>DrvItemContentChanged</u>	<b>Informationsbereich für geänderte Inhalte</b>
ContentChanged	<u>logic</u>	<b>Datei wurde geändert (true)</b>
ItemFileCustom	<u>DrvItemFileCustom</u>	<b>Informationsbereich für benutzerdefinierte Daten (optional)</b>
Custom01	<u>alpha(4096)</u>	<b>Benutzerdefinierte Informationen</b>
ItemFileHash	<u>DrvItemFileHash</u>	<b>Informationsbereich der Prüfsumme</b>
FileHash	<u>alpha(250)</u>	<b>Hash der gespeicherten Datei</b>
ItemFileContentDisk	<u>DrvItemFileContentDisk</u>	<b>Informationsbereich für den Inhalt der Datei</b>
FilePath	<u>alpha(1024)</u>	<b>Pfad und Dateiname der Datei im temporären Pfad</b>
ItemFileSize	<u>DrvItemFileSize</u>	<b>Informationsbereich der Dateigröße (optional)</b>
FileSize	<u>bigint</u>	<b>Größe der Datei (in Bytes)</b>
ItemFileAttributes	<u>DrvItemFileAttributes</u>	<b>Informationsbereich für Dateiattribute (optional)</b>
FileAttributes	<u>int</u>	<b>Dateiattribute</b>
ItemFileTime	<u>DrvItemFileTime</u>	<b>Informationsbereich für Dateidatum und -uhrzeit (optional)</b>
CreationTime	<u>bigint</u>	<b>Zeitstempel der Erzeugung der Datei</b>
LastAccessTime	<u>bigint</u>	<b>Zeitstempel des letzten Zugriffs</b>
LastWriteTime	<u>bigint</u>	<b>Zeitstempel der letzten Änderung</b>

## Kontakt

<u>ItemFileAuthorisation</u> _DrvItemFileAuthorisation		Informationsbereich für Berechtigungen (optional)
UserId	<u>int</u>	Zugriffs-Id
Authorisation	<u>int</u>	Berechtigungen
ItemResult	<u>DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</u>	Ergebniswert

Die Bereiche **ItemFileCustom**, **ItemFileSize**, **ItemFileAttributes**, **ItemFileTime** und **ItemFileAuthorisation** sind optional. Die Informationen werden nur dann geändert, wenn diese Bereiche im Antwortdatenpaket enthalten ist. Zum Löschen der Daten muss ein Leerstring oder 0 (für die int-Werte) angegeben werden. Bei **ItemFileTime** muss immer ein gültiger Zeitstempel angegeben werden, wenn dieser gesetzt wird.

Die Bereiche **ItemFileHash** und **ItemFileContentDisk** müssen nur dann angegeben werden, wenn sich der Inhalt der Datei geändert hat, also im Antwortpaket **ContentChanged** auf true gesetzt ist. Das **ItemFileHash** kann beim nächsten Aufruf von DrvOpenFile und DrvReadFile verwendet werden, um zu prüfen, ob die Datei verändert wurde.

Die Datei wird nur dann gelesen, wenn in **Result** ErrOk (0) angegeben wird. Durch das Setzen eines ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

Beispiel:

```
if (tDrvRequestMessageId = _DrvReqReadFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSck
```

```
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## Kontakt

### DrvOpenFile

Aufruf beim Öffnen einer Datei

#### Request

MessageId	<u>DrvReqOpenFile</u>	Id des Ereignisses
ItemHeader	<u>DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

**ItemAuthUserDomain** DrvItemAuthUserDomain Betriebssystem angemeldeten Benutzers

UserName	<u>alpha(80)</u>	Name des angemeldeten Benutzers
DomainName	<u>alpha(80)</u>	Domäne des angemeldeten Benutzers
ItemFilePath	<u>DrvItemFilePath</u>	Informationsbereich für die Datei Pfad und Dateiname (ohne Laufwerksbuchstabe)
FilePath	<u>alpha(1024)</u>	Informationsbereich für benutzerdefinierte Daten
ItemFileCustom	<u>DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Informationen
Custom01	<u>alpha(4096)</u>	Informationsbereich der Prüfsumme
ItemFileHash	<u>DrvItemFileHash</u>	Prüfsumme der Datei
FileHash	<u>alpha(250)</u>	Informationsbereich für die Cache-Nutzung
ItemCacheTime	<u>DrvItemCacheTime</u>	0 oder der Zeitpunkt, zu dem die Datei zuletzt abgefragt wurde
CacheTime	<u>bint</u>	Informationsbereich für die Zugriffsart
ItemFileAccess	<u>DrvItem FileAccess</u>	Zugriffsart (siehe Text)
FsiFlags	<u>int</u>	

Siehe Verwandte Befehle

Dieses Ereignis wird aufgerufen, wenn eine Datei auf dem Laufwerk mit Lese-und/oder Schreibzugriff geöffnet wird. In dem Request wird der am Betriebssystem angemeldete Benutzer übergeben. In FilePath befindet sich der Pfad (ohne

## Kontakt

Laufwerksbuchstaben) und der Dateiname der geöffneten Datei. Wurden der Datei benutzerspezifische Informationen zugeordnet, befinden sich diese in Custom01. In FileHash wird die Prüfsumme der Datei übertragen, die beim letzten Aufruf von DrvReadFile gesetzt wurde. Befindet sich die Datei bereits im Cache, wird in CacheTime der Zeitpunkt angegeben, zu dem die Datei zuletzt mit DrvReadFile abgefragt wurde. Befindet sich die Datei nicht im Cache, wird 0 übergeben. Der Inhalt von FsiFlags kann mit folgenden Konstanten, oder Kombinationen davon, verglichen werden:

<u>FsiAcsR</u>	nur Lesezugriff
<u>FsiAcsW</u>	nur Schreibzugriff
<u>FsiAcsRW</u>	Lese- und Schreibzugriff
<u>FsiDenyNone</u>	kein exklusiver Zugriff
<u>FsiStdRead</u>	Standard-Lesemodus
<u>FsiStdWrite</u>	Standard-Schreibmodus

Über die Prüfsumme oder den Cache-Zeitpunkt kann der Programmierer entscheiden, ob sich die Datei in der Zwischenzeit geändert hat und neu übertragen werden muss. Ist das der Fall, muss in der Response im Item ItemContentChanged der Wert true übermittelt werden. Anschließend wird das Ereignis DrvReadFile ausgelöst, um den Dateinhalt neu zu übertragen.

## Response

<b>MessageId</b>	<u>DrvResOpenFile</u>	<b>Id der Antwort</b>
<b>ItemHeader</b>	<u>DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
<b>ProtocolId</b>	<u>0xF3A49E52</u>	<b>Id des verwendeten Protokolls</b>
<b>ProtocolVersion</b>	<u>0x00040000</u>	<b>Version des verwendeten Protokolls</b>
<b>ItemFileCustom</b>	<u>DrvItemFileCustom</u>	<b>Informationsbereich für benutzerdefinierte Daten (optional)</b>
<b>Custom01</b>	<u>alpha(4096)</u>	<b>Benutzerdefinierte Informationen</b>
<b>ItemFileSize</b>	<u>DrvItemFileSize</u>	<b>Informationsbereich der Dateigröße (optional)</b>
<b>FileSize</b>	<u>bigint</u>	<b>Größe der Datei (in Bytes)</b>
<b>ItemFileAttributes</b>	<u>DrvItemFileAttributes</u>	<b>Informationsbereich für Dateiattribute (optional)</b>
<b>FileAttributes</b>	<u>int</u>	<b>Dateiattribute</b>
<b>ItemFileTime</b>	<u>DrvItemFileTime</u>	<b>Informationsbereich für Dateidatum und -uhrzeit (optional)</b>
<b>CreationTime</b>	<u>bigint</u>	<b>Zeitstempel der Erzeugung der Datei</b>
<b>LastAccessTime</b>	<u>bigint</u>	<b>Zeitstempel des letzten Zugriffs</b>
<b>LastWriteTime</b>	<u>bigint</u>	<b>Zeitstempel der letzten Änderung</b>
<b>ItemFileAuthorisation</b>	<u>DrvItemFileAuthorisation</u>	<b>Informationsbereich für Berechtigungen (optional)</b>
<b>UserId</b>	<u>int</u>	<b>Zugriffs-ID</b>
<b>Authorisation</b>	<u>int</u>	<b>Berechtigungen</b>
<b>ItemContentChanged</b>	<u>DrvItemContentChanged</u>	<b>Informationsbereich für geänderte</b>

## Kontakt

<b>ContentChanged</b>	<u>logic</u>	<b>Inhalte</b>
<b>ItemResult</b>	<u>DrvItemResult</u>	<b>Datei wurde geändert (true)</b>
<b>Result</b>	<u>int</u>	<b>Ergebnis-Bereich</b>

Die Bereiche ItemFileCustom, ItemFileSize, ItemFileAttributes, ItemFileType und ItemFileAuthorisation sind optional. Die Informationen werden nur dann geändert, wenn diese Bereiche im Antwortdatenpaket enthalten sind. Zum Löschen der Daten muss ein Leerstring oder 0 (für die int-Werte) angegeben werden. Bei ItemFileType muss immer ein gültiger Zeitstempel angegeben werden, wenn dieser gesetzt wird.

Der Bereich ItemContentChanged muss zurückgegeben werden. Dieses Item bestimmt, ob die Datei geändert wurde. Das Ereignis DryReadFile wird nur ausgelöst, wenn true übertragen wird. Mit Hilfe der Items ItemCacheTime und ItemFileHash aus dem Request kann ermittelt werden, ob in dem Bereich true oder false übertragen werden muss.

Die Datei wird nur dann geöffnet, wenn in Result ErrOk (0) angegeben wird. Durch das Setzen eines ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

Beispiel:

```
if (tDrvRequestMessageId = _DrvReqOpenFile){  tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSck
```

```
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time  tDrvMsxWrite->MsxWrite(_M
```

## Kontakt

### **DrvCloseFile**

Aufruf beim Schließen einer Datei

#### Request

<b>MessageId</b>	<u>DrvReqCloseFile</u>	<b>Id des Ereignisses</b>
<b>ItemHeader</b>	<u>DrvItemHeader</u>	<b>Kopf des Datenpakets</b>
<b>ProtocolId</b>	<b>0xF3A49E52</b>	<b>Id des verwendeten Protokolls</b>
<b>ProtocolVersion</b>	<b>0x00040000</b>	<b>Version des verwendeten Protokolls</b>

### **ItemAuthUserDomain \_DrvItemAuthUserDomain Betriebssystem**

<b>UserName</b>	<u>alpha(80)</u>	<b>Benutzername des angemeldeten Benutzers</b>
<b>DomainName</b>	<u>alpha(80)</u>	<b>Domäne des angemeldeten Benutzers</b>
<b>ItemFilePath</b>	<u>DrvItemFilePath</u>	<b>Informationsbereich für die Datei Pfad und Dateiname (ohne Laufwerksbuchstabe)</b>
<b>FilePath</b>	<u>alpha(1024)</u>	<b>Informationsbereich für benutzerdefinierte Daten</b>
<b>ItemFileCustom</b>	<u>DrvItemFileCustom</u>	<b>Informationsbereich für benutzerdefinierte Informationen</b>
<b>Custom01</b>	<u>alpha(4096)</u>	<b>Informationsbereich für Dateiattribute</b>
<b>ItemFileAttributes</b>	<u>DrvItemFileAttributes</u>	<b>Dateiattribute</b>
<b>FileAttributes</b>	<u>int</u>	<b>Informationsbereich für Dateidatum und -uhrzeit</b>
<b>ItemFileTime</b>	<u>DrvItemFileTime</u>	<b>Datum und Uhrzeit der Dateierzeugung</b>
<b>CreationTime</b>	<u>bigint</u>	<b>Datum und Uhrzeit des letzten Zugriffs</b>
<b>LastAccessTime</b>	<u>bigint</u>	<b>Datum und Uhrzeit der letzten Änderung</b>
<b>LastWriteTime</b>	<u>bigint</u>	
<b>ItemContentChanged _DrvItemContentChanged</b>		<b>Informationsbereich für geänderte Inhalte Datei wurde geändert (true)</b>
<b>ContentChanged</b>	<u>logic</u>	<b>Informationsbereich für den Inhalt der</b>
<b>ItemFileContentDisk</b>	<u>DrvItemFileContentDisk</u>	

## Kontakt

### Datei

#### Pfad und Dateiname

alpha(1024) der Datei im temporären Pfad

Siehe

#### Verwandte Befehle

Das Ereignis wird ausgelöst, wenn die zuvor geöffnete Datei wieder geschlossen wird.

Im Datenpaket befinden sich folgende Informationen:

#### ItemAuthUserDomain

Hier wird der beim Betriebssystem angemeldete Benutzer und die Domäne, in der er angemeldet ist, übergeben.

#### ItemFilePath

Der Pfad und Dateiname der geschlossenen Datei. Der Pfad wird ohne den Laufwerksbuchstaben angegeben.

#### ItemFileCustom

Wurden für die Datei benutzerdefinierte Daten angegeben, befinden sich diese in diesem Bereich.

#### ItemFileAttributes

Der Wert in dem Bereich FileAttributes kann mit folgenden Konstanten verglichen werden:

#### FsiAttrHidden Versteckte Datei

#### FsiAttrSystem Systemdatei

#### FsiAttrDir Verzeichnis

#### FsiAttrArchive Archivdatei

#### ItemFileTime

In diesem Bereich befinden sich das Datum und die Uhrzeit für die Erstellung (CreationTime), den letzten Zugriff (LastAccessTime) und die letzte Änderung (LastWriteTime) der Datei. Der Wert wird als Zeitstempel angegeben, dieser kann mit folgender Anweisung abgefragt werden:

```
tDrvMsxRead->MsxRead(_MsxData, tTimeStamp); tCalTime # CnvCB(tTimeStamp);
```

#### ItemContentChanged

Wurde der Inhalt der Datei geändert, wird in ContentChanged true übergeben. Hat keine Veränderung statt gefunden, ist der Wert false.

#### ItemFileContentDisk

Der Inhalt der Datei wird in einem temporären Verzeichnis gespeichert. Das Verzeichnis und der Dateiname dieser Datei befinden sich in FilePath. Die Datei darf in dem Pfad nicht gelöscht oder geändert werden. Die Werte der Datei (Hash, Dateigröße, ...) müssen in dem Antwortpaket übertragen werden.

## Response