

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(..., \_JobMxTimeoutRead, ...) 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) { tMxHdlR->MxRead(_MxItem, tMsgItem);
{ ... try { // Job starten tJobID # JobStart(_JobThread, 0, 'Service:JobWork'); tJ
```


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.

## 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 zugehörigen Ereignisfunktion übergeben.

In den Eigenschaften des Objekts sind unter anderem Informationen zu dem aufrufenden Task, Statusinformationen und den verwendeten Ressourcen abgelegt.

## 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.

## 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 entsprechendes 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.

### 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.

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.

### 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

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 Datenpakets 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, (tReques
```

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.

\_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)**

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

- **\_DrvItemFileAccess (1062)**

Informationsbereich für den Zugriffsmodus auf eine Datei.

int Zugriffs-Flags

- **\_DrvItemFileAttributes (1071)**

Informationsbereich für die Dateiattribute.

int Dateiattribute

- **\_DrvItemFileAuthorisation (1064)**

Informationsbereich für Zugriffsrechte.

int Zugriffs-Id

int Berechtigungen

Die Berechtigung kann als Kombination der folgenden Konstanten angegeben werden:

_DrvAuthAll	Alle Rechte vorhanden
_DrvAuthNone	Keine Rechte vorhanden
_DrvAuthList	Dateien und Verzeichnisse listen
_DrvAuthDelete	Dateien und Verzeichnisse löschen
_DrvAuthRename	Dateien und Verzeichnisse umbenennen
_DrvAuthMove	Dateien und Verzeichnisse verschieben
_DrvAuthSetAttributes	Attribute von Dateien und Verzeichnissen setzen
_DrvAuthRead	Datei lesen
_DrvAuthWrite	Datei schreiben
_DrvAuthExecute	Datei ausführen
_DrvAuthCreateDirectory	Verzeichnis erstellen
_DrvAuthCreateFile	Datei erstellen

• **\_DrvItemFileContentDisk (1065)**

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

- **\_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 Laufwerksnamen.

alpha Name des Laufwerks

- **\_DrvItemVolumeSize (1040)**

Informationsbereich für die Laufwerksgröße und des freien Speicherplatzes.

bigint freier Speicherbereich in Bytes

bigint gesamte Speicherkapazität in Bytes

### 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

## DrvInit

Aufruf beim Start des Laufwerkstreibers

### Request

MessageId	<u>_DrvReqInit</u>	Id des Ereignisses
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemComputerName	<u>_DrvItemComputerName</u>	Informationsbereich des Computers
ComputerName	<u>alpha</u> (80)	Name des Rechners auf dem der Laufwerkstreiber gestartet ist
ItemDriverData	<u>_DrvItemDriverData</u>	Informationsbereich des Treibers
DriverData	<u>alpha</u> (80)	Versionsnummer des installierten Treibers

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

MessageId	<u>_DrvResInit</u>	Id der Antwort
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemCfgMountLocal	<u>_DrvItemCfgMountLocal</u>	Informationsbereich des lokalen Laufwerks
DriveLetter	<u>alpha</u> (1)	Der Laufwerksbuchstabe des lokalen Laufwerks
ItemCfgMountNetwork	<u>_DrvItemCfgMountNetwork</u>	Informationsbereich für Netzwerkfreigaben
ShareName	<u>alpha</u> (80)	Freigabename des Laufwerks
ItemCfgCacheDisk	<u>_DrvItemCfgCacheDisk</u>	Informationsbereich für den Cache





CacheMaxSize	<u>bigint</u>	Maximaler Festplattenspeicher, der zur Zwischenspeicherung der Inhalte verwendet wird. Es wird ein entsprechendes Verzeichnis im temporären Pfad des Betriebssystems angelegt.
CachePath	<u>alpha</u> (1024)	Laufwerk und Pfad für temporäre Dateien
ItemCfgIgnoreMaskRequest	<u>_DrvItemCfgIgnoreMaskRequest</u>	Informationsbereich für Dateifilter Hier kann eine Pipe-separierte Liste mit Dateimasken angegeben werden.
FileMask	<u>alpha</u> (4096)	Für die entsprechenden Dateien werden keine Dateiinformationen (siehe <u>DrvGetFileInfo</u> ) abgefragt.
ItemResult	<u>_DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</u>	Ergebniswert

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 Netzwerkfreigabe nicht erzeugt werden, wird ein Fehler generiert und das Ereignis DrvError aufgerufen.

### Beispiel:

```
if (tDrvRequestId = _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');
```

DrvTerm

Aufruf beim Beenden des Laufwerkstreiber

### Request

MessageId	_DrvReqTerm	Id des Ereignisses
ItemHeader	_DrvItemHeader	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls

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	_DrvResTerm	Id der Antwort
ItemHeader	_DrvItemHeader	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemResult	<a href="#">_DrvItemResult</a>	Ergebnis-Bereich
Result	<a href="#">int</a>	Ergebniswert

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);
```

**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>_DrvItemResult</u>	Informationsbereich für das Resultat
Result	<u>int</u>	Fehlerwert
ItemErrorText	<u>_DrvItemErrorText</u>	Informationsbereich des Fehlertextes
ErrorText	<u>alpha</u> (250)	Fehlertext

Siehe Verwandte Befehle

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:

<u>_ErrDrvDriverNotInstalled</u>	Der Grätetreiber des Laufwerkstreibers ist nicht installiert.
<u>_ErrDrvInitFailed</u>	Fehler bei der Initialisierung des Laufwerkstreibers.
<u>_ErrDrvInitInvalidData</u>	Kein Laufwerksbuchstabe und Freigabename angegeben.
<u>_ErrDrvInitDriveLetterInUse</u>	Der Laufwerksbuchstabe ist in Benutzung.
<u>_ErrDrvCreateTempPath</u>	Temporärer Pfad ist ungültig oder konnte nicht angelegt werden.
<u>_ErrDrvCreateTempFile</u>	Temporäre Datei konnte nicht angelegt werden.
<u>_ErrDrvClearTempPath</u>	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
```

DrvLoginDomainUser

Aufruf beim Anmelden eines Benutzers

### Request

MessageId	<u>_DrvReqLoginDomainUser</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 des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Benutzername des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domain des angemeldeten Benutzers
ItemAuthAccessId	<u>_DrvItemAuthAccessId</u>	Informationsbereich des Benutzers
AccessId	<u>int</u>	Zugriffs-Id des Benutzers

Siehe Verwandte Befehle

Dieses Ereignis wird bei der Anmeldung eines Benutzers ausgelöst und wenn eine erneute Abfrage des Benutzers erfolgt. Erfolgt die erneuert Abfrage aufgrund des zuvor angegebenen Timeouts, wird die bisherige AccessId im Datenpaket des Requests mit übergeben.

### Response

MessageId	<u>_DrvResLoginDomainUser</u>	Id der Antwort
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemAuthAccessId	<u>_DrvItemAuthAccessId</u>	Informationsbereich des Benutzers
AccessId	<u>int</u>	Zugriffs-Id des Benutzers
ItemTimeout	<u>_DrvItemTimeout</u>	Informationsbereich für den Timeout
Timeout	<u>int</u>	Zeit bis zur erneuten Abfrage des Benutzers
ItemResult	<u>_DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</u>	Ergebniswert

Wird in Result ein Wert ungleich 0 (\_ErrOk) angegeben, wird der Benutzer abgewiesen. Er kann dann auf dem Laufwerk keine Aktionen durchführen.

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 (tDrvRequestId = _DrvReqLoginDomainUser){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite
```

## DrvGetVolumeData

Aufruf bei der Anforderung der Laufwerksdaten

**Request**

MessageId	<u>_DrvReqGetVolumeData</u>	Id des Ereignisses
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	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	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemVolumeSize	<u>_DrvItemVolumeSize</u>	Informationsbereich für den Speicherplatz
FreeSize	<u>bigint</u>	Freier Speicherplatz auf dem Laufwerk (in Bytes)
TotalSize	<u>bigint</u>	Gesamtkapazität des Laufwerks (in Bytes)
ItemVolumeId	<u>_DrvItemVolumeId</u>	Informationsbereich für die Id des Laufwerks
VolumeId	<u>int</u>	Id des Laufwerks
ItemVolumeLabel	<u>_DrvItemVolumeLabel</u>	Informationsbereich des Laufwerksnamens.
VolumeLabel	<u>alpha(80)</u>	Name des Laufwerks.
ItemTimeout	<u>_DrvItemTimeout</u>	Informationsbereich für den Timeout
Timeout	<u>int</u>	Zeit bis zum erneuten Abruf der Informationen (in Millisekunden).

**Beispiel:**

```
if (tDrvRequestMessageId = _DrvReqGetVolumeData){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite,
```

## 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</u> (80)	Name des Benutzers
DomainName	<u>alpha</u> (80)	Name der Domain
ItemCacheTime	<u>_DrvItemCacheTime</u>	Informationsbereich für die Cache-Nutzung
CacheTime	<u>bigint</u>	0 oder der Zeitpunkt, zu dem der Eintrag zuletzt abgefragt wurde
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich des Verzeichnisses
FilePath	<u>alpha</u> (1024)	Pfad des Verzeichnisses (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für den Programmierer
Custom01	<u>alpha</u> (4096)	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 bigint gewandelten jetzigen Zeitpunkt abgezogen werden, um das Alter des Cache-Eintrags zu ermitteln.

```
tCaltmeNow->vmSystemTime();tNow # CnvBC(tCaltmeNow);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.



**Response**

MessageId	<u>_DrvResGetFileEntries</u>	Id der Antwort
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemContentChanged	<u>_DrvItemContentChanged</u>	Informationsbereich für Änderungen
ContentChanged	<u>logic</u>	Inhalt wurde geändert ( <u>true</u> )
ItemTimeout	<u>_DrvItemTimeout</u>	Informationsbereich für den Timeout
Timeout	<u>int</u>	Maximale Zeit bis zur nächsten Abfrage des Verzeichnissesinhaltes (in Millisekunden)
ItemResult	<u>_DrvItemResult</u>	Informationsbereich für das Ergebnis
Timeout	<u>int</u>	Bei einem Resultat ungleich 0 ( <u>_ErrOk</u> ) werden keine Verzeichniseinträge übertragen

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 Verzeichnissesinhaltes, 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:

**Response**

MessageId	<u>_DrvResGetFileEntryData</u>	Id der Antwort
ItemResult	<u>_DrvItemResult</u>	Informationsbereich des Ergebnisses
Result	<u>int</u>	Ergebnis der Operation
ItemFileName	<u>– _DrvItemFileName</u>	Informationsbereich für Dateinamen
FileName	<u>alpha</u> (1024)	Dateiname
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für den Programmierer
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileSize	<u>_DrvItemFileSize</u>	Informationsbereich für die Dateigröße
FileSize	<u>bigint</u>	Größe der Datei (in Bytes)
ItemFileAttributes	<u>_DrvItemFileAttributes</u>	Informationsbereich für die Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>_DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit
CreationTime	<u>bigint</u>	

		Datum und Uhrzeit der Dateierstellung
LastAccessTime	<u>bigint</u>	Datum und Uhrzeit des letzten Dateizugriffs
LastWriteTime	<u>bigint</u>	Datum und Uhrzeit der letzten Änderung
ItemFileSubData	<u>DrvItemFileSubData</u>	Informationsbereich für weitere Daten
Empty	<u>logic</u>	Die Datei ist leer ( <u>true</u> )
ItemFileAuthorisation	<u>DrvItemFileAuthorisation</u>	Informationsbereich für die Dateiberechtigungen
UserId	<u>int</u>	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.

- **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>_DrvAuthCreateDirectory</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 (tDrvRequestId = _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->MsxW
```

## Kontakt

### DrvGetFileInfo

Aufruf beim Abfragen von Dateiinformatioren

#### 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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha</u> (1024)	Pfad- und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen

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</u> (4096)	Benutzerdefinierte Informationen

ItemFileSize	<u>DrvItemFileSize</u>	Informationsbereich für die Dateigröße
FileSize	<u>bigint</u>	Größe der Datei (in Bytes)
ItemFileAttributes	<u>DrvItemFileAttributes</u>	Informationsbereich für die Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit
CreationTime	<u>bigint</u>	Datum und Uhrzeit der Dateierstellung
LastAccessTime	<u>bigint</u>	Datum und Uhrzeit des letzten Dateizugriffs
LastWriteTime	<u>bigint</u>	Datum und Uhrzeit der letzten Änderung
ItemFileSubData	<u>DrvItemFileSubData</u>	Informationsbereich für weitere Daten
Empty	<u>logic</u>	Das Verzeichnis ist leer ( <u>true</u> ).
ItemFileAuthorisation	<u>DrvItemFileAuthorisation</u>	Informationsbereich für die Dateiberechtigungen
AccessId	<u>int</u>	Zugriffs-Id
Authorisation	<u>int</u>	Berechtigungen an der Datei (siehe Text)
ItemResult	<u>DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</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>_DrvAuthCreateDirectory</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->MsxW
```

## 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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich für die Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileAccess	<u>_DrvItemFileAccess</u>	Informationsbereich für die Zugriffsart
FsiFlags	<u>int</u>	Zugriffsart (siehe Text)
ItemFileAttributes	<u>_DrvItemFileAttributes</u>	Informationsbereich für Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute

Siehe Verwandte Befehle

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:

\_FsiAttrHidden Versteckte Datei  
\_FsiAttrSystem Systemdatei  
\_FsiAttrDir Verzeichnis  
\_FsiAttrArchive Archivdatei

### Response

MessageId	<u>_DrvResCreateFile</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
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileAuthorisation	<u>_DrvItemFileAuthorisation</u>	Informationsbereich für Berechtigungen
UserId	<u>int</u>	Zugriffs-Id
Authorisation	<u>int</u>	Berechtigungen
ItemResult	<u>_DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</u>	Ergebniswert

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>	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>_DrvAuthCreateDirectory</u>	Verzeichnis erstellen
<u>_DrvAuthCreateFile</u>	Datei erstellen

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
```



**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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha</u> (1024)	Path- und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen

Siehe [Verwandte Befehle](#)

Dieses Ereignis wird aufgerufen, wenn eine Datei gelöscht wird. Die zu löschende 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	<u>int</u>	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
```

**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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha</u> (1024)	Pfad- und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFilePathNew	<u>_DrvItemFilePathNew</u>	Informationsbereich des neuen Pfades
FilePathNew	<u>alpha</u> (1024)	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</u> (4096)	Benutzerdefinierte Informationen
ItemFileSize	<u>_DrvItemFileSize</u>	Informationsbereich der Dateigröße

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 (tDrvRequestId = _DrvReqMoveFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSch
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## 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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileNameNew	<u>_DrvItemFileNameNew</u>	Informationsbereich des neuen Dateinamens
FileNameNew	<u>alpha</u> (1024)	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</u> (4096)	Benutzerdefinierte Informationen

ItemFileSize	<u>__DrvItemFileSize</u>	Informationsbereich der Dateigröße (optional)
FileSize	<u>bigint</u>	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 umbenannt, wenn in Result \_\_ErrOk (0) angegeben wird. Durch das Setzen eines \_\_ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

### Beispiel:

```
if (tDrvRequestId = _DrvReqRenameFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tS
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## 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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich der Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileHash	<u>_DrvItemFileHash</u>	Informationsbereich der Prüfsumme
FileHash	<u>alpha</u> (250)	Prüfsumme der Datei
ItemCacheTime	<u>_DrvItemCacheTime</u>	Informationsbereich für die Cache-Nutzung
CacheTime	<u>bigint</u>	0 oder der Zeitpunkt, zu dem die Datei zuletzt abgefragt wurde
ItemTempPath	<u>_DrvItemTempPath</u>	Informationsbereich für den temporären Pfad
TempPath	<u>alpha</u> (1024)	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

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>	Id der Antwort
ItemHeader	<u>_DrvItemHeader</u>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemContentChanged	<u>_DrvItemContentChanged</u>	Informationsbereich für geänderte Inhalte
ContentChanged	<u>logic</u>	Datei wurde geändert ( <u>true</u> )
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten (optional)
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileHash	<u>_DrvItemFileHash</u>	Informationsbereich der Prüfsumme
FileHash	<u>alpha</u> (250)	Hash der gespeicherten Datei
ItemFileContentDisk	<u>_DrvItemFileContentDisk</u>	Informationsbereich für den Inhalt der Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname der Datei im temporären Pfad
ItemFileSize	<u>_DrvItemFileSize</u>	Informationsbereich der Dateigröße (optional)
FileSize	<u>bigint</u>	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

## Kontakt

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 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, tSch
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```



## 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	<u>_DrvItemAuthUserDomain</u>	Informationsbereich des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich für die Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileHash	<u>_DrvItemFileHash</u>	Informationsbereich der Prüfsumme
FileHash	<u>alpha</u> (250)	Prüfsumme der Datei
ItemCacheTime	<u>_DrvItemCacheTime</u>	Informationsbereich für die Cache-Nutzung
CacheTime	<u>bigint</u>	0 oder der Zeitpunkt, zu dem die Datei zuletzt abgefragt wurde
ItemFileAccess	<u>_DrvItemFileAccess</u>	Informationsbereich für die Zugriffsart
FsiFlags	<u>int</u>	Zugriffsart (siehe Text)

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

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 Dateiinhalt neu zu übertragen.

## Response

MessageId	<u>DrvResOpenFile</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</u> (4096)	Benutzerdefinierte Informationen
ItemFileSize	<u>DrvItemFileSize</u>	Informationsbereich der Dateigröße (optional)
FileSize	<u>bigint</u>	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
ItemContentChanged	<u>DrvItemContentChanged</u>	Informationsbereich für geänderte

## Kontakt

ContentChanged	<u>logic</u>	Inhalte
ItemResult	<u>DrvItemResult</u>	Datei wurde geändert ( <u>true</u> )
Result	<u>int</u>	Ergebnis-Bereich
		Ergebniswert

Die Bereiche ItemFileCustom, ItemFileSize, ItemFileAttributes, ItemFileTime 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 ItemFileTime 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 DrvReadFile 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 (tDrvRequestId = _DrvReqOpenFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSch
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## DrvCloseFile

Aufruf beim Schließen einer Datei

### Request

MessageId	<u>_DrvReqCloseFile</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 des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich für die Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileAttributes	<u>_DrvItemFileAttributes</u>	Informationsbereich für Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>_DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit
CreationTime	<u>bigint</u>	Datum und Uhrzeit der Dateierzeugung
LastAccessTime	<u>bigint</u>	Datum und Uhrzeit des letzten Zugriffs
LastWriteTime	<u>bigint</u>	Datum und Uhrzeit der letzten Änderung
ItemContentChanged	<u>_DrvItemContentChanged</u>	Informationsbereich für geänderte Inhalte
ContentChanged	<u>logic</u>	Datei wurde geändert ( <u>true</u> )
ItemFileContentDisk	<u>_DrvItemFileContentDisk</u>	Informationsbereich für den Inhalt der

Datei  
Pfad und Dateiname  
der Datei im  
temporären Pfad

FilePath alpha(1024)

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

## Kontakt

MessageId	<u>_DrvResCloseFile</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
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileHash	<u>_DrvItemFileHash</u>	Informationsbereich der Prüfsumme
FileHash	<u>alpha</u> (250)	Hash der gespeicherten Datei
ItemFileSize	<u>_DrvItemFileSize</u>	Informationsbereich der Dateigröße
FileSize	<u>bigint</u>	Größe der Datei (in Bytes)
ItemFileAttributes	<u>_DrvItemFileAttributes</u>	Informationsbereich für Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>_DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit
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
UserId	<u>int</u>	Zugriffs-Id
Authorisation	<u>int</u>	Berechtigungen

Der Bereich ItemFileCustom ist optional. Die benutzerdefinierten Informationen werden nur dann geändert, wenn dieser Bereich im Antwortdatenpaket enthalten ist. Zum Löschen der benutzerdefinierten Daten muss ein Leerstring angegeben werden.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqCloseFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSc
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## DrvFlushFile

Aufruf beim Schreiben des Festplattencaches

**Request**

MessageId	<u>_DrvReqFlushFile</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 des am Betriebssystem angemeldeten Benutzers
UserName	<u>alpha</u> (80)	Name des angemeldeten Benutzers
DomainName	<u>alpha</u> (80)	Domäne des angemeldeten Benutzers
ItemFilePath	<u>_DrvItemFilePath</u>	Informationsbereich für die Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname (ohne Laufwerksbuchstabe)
ItemFileCustom	<u>_DrvItemFileCustom</u>	Informationsbereich für benutzerdefinierte Daten
Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileAttributes	<u>_DrvItemFileAttributes</u>	Informationsbereich für Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>_DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit
CreationTime	<u>bigint</u>	Datum und Uhrzeit der Dateierzeugung
LastAccessTime	<u>bigint</u>	Datum und Uhrzeit des letzten Zugriffs
LastWriteTime	<u>bigint</u>	Datum und Uhrzeit der letzten Änderung
ItemFileContentDisk	<u>_DrvItemFileContentDisk</u>	Informationsbereich für den Inhalt der Datei
FilePath	<u>alpha</u> (1024)	Pfad und Dateiname der Datei im temporären Pfad

Siehe [Verwandte Befehle](#)

Das Ereignis wird ausgelöst, wenn die Datei aus dem Festplattencache auf das Laufwerk geschrieben wird. Die Datei bleibt weiterhin geöffnet. Das Schließen der Datei erfolgt in [DrvCloseFile](#). 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 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 gelesen werden:

```
tDrvMsxRead->MsxRead(_MsxData, tTimeStamp);tCalTime # CnvCB(tTimeStamp);
```

- **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

MessageId	<a href="#">_DrvResFlushFile</a>	Id der Antwort
ItemHeader	<a href="#">_DrvItemHeader</a>	Kopf des Datenpakets
ProtocolId	0xF3A49E52	Id des verwendeten Protokolls
ProtocolVersion	0x00040000	Version des verwendeten Protokolls
ItemFileCustom	<a href="#">_DrvItemFileCustom</a>	Informationsbereich für benutzerdefinierte Daten



Custom01	<u>alpha</u> (4096)	Benutzerdefinierte Informationen
ItemFileHash	<u>_DrvItemFileHash</u>	Informationsbereich der Prüfsumme
FileHash	<u>alpha</u> (250)	Hash der gespeicherten Datei
ItemFileSize	<u>_DrvItemFileSize</u>	Informationsbereich der Dateigröße
FileSize	<u>bigint</u>	Größe der Datei (in Bytes)
ItemFileAttributes	<u>_DrvItemFileAttributes</u>	Informationsbereich für Dateiattribute
FileAttributes	<u>int</u>	Dateiattribute
ItemFileTime	<u>_DrvItemFileTime</u>	Informationsbereich für Dateidatum und -uhrzeit
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
UserId	<u>int</u>	Zugriffs-Id
Authorisation	<u>int</u>	Berechtigungen
ItemResult	<u>_DrvItemResult</u>	Ergebnis-Bereich
Result	<u>int</u>	Ergebniswert

Der Bereich ItemFileCustom ist optional. Die benutzerdefinierten Informationen werden nur dann geändert, wenn dieser Bereich im Antwortdatenpaket enthalten ist. Zum Löschen der benutzerdefinierten Daten muss ein Leerstring angegeben werden.

Die Datei wird nur dann geschrieben, wenn in Result ErrOk (0) angegeben wird. Durch das Setzen eines ErrFsi...-Wertes können andere Fehlerzustände übermittelt werden.

### Beispiel:

```
if (tDrvRequestMessageId = _DrvReqFlushFile){ tDrvMsxWrite # MsxOpen(_MsxSocket | _MsxWrite, tSc
    tDrvMsxWrite->MsxWrite(_MsxItem, _DrvItemFileTime); // set file time tDrvMsxWrite->MsxWrite(_M
```

## CONZEPT 16-Laufwerkstreiber - Referenzimplementation

Beschreibung einer Beispielimplementation für den Laufwerkstreiber

Alle Prozeduren der Referenzimplementation sind in der Beispiel-Datenbank

"CodeLibrary" enthalten. Die Prozeduren können aus dieser Datenbank exportiert und eine beliebige andere Datenbank importiert werden. Weitere Ressourcen werden nicht benötigt.

In diesem Abschnitt wird die Referenzimplementation beschreiben. Die Beschreibung unterteilt sich in folgende Abschnitte:

- Aufgabenstellung
- Vorbereitung der Anwendung
- Arbeitsweise der Implementation

Referenzimplementation - Aufgabenstellung

Beschreibung der Aufgabenstellung für die Referenzimplementation des Laufwerkstreibers

In der Referenzimplementation sind folgende Funktionen realisiert:

- **Abbildung der BLObs in einem Laufwerk**

Alle BLObs (Binary Large Objects) innerhalb der Datenbank sollen in einem Laufwerk angezeigt werden können.

- **Abbildung der Verzeichnisstruktur der BLObs**

Die Verzeichnisstrukturen, in denen die BLObs in der Datenbank organisiert sind, sollen sich als Verzeichnisse in dem Laufwerk widerspiegeln.

- **Dateioperationen auf dem Laufwerk entsprechen den Operationen in der Datenbank**

Operationen auf dem Laufwerk sollen eine Entsprechung in der Datenbank besitzen. Folgende Dateioperationen sollen abgebildet werden:

- ♦ **Anlegen** Neu angelegte Dateien sollen als BLOb in die Datenbank importiert werden.
- ♦ **Kopieren** Werden auf dem Laufwerk Dateien kopiert, werden diese auch in der Datenbank in das entsprechende Verzeichnis kopiert.
- ♦ **Verschieben** Werden auf dem Laufwerk Dateien verschoben, werden diese auch in der Datenbank in das entsprechende Verzeichnis verschoben.
- ♦ **Löschen** Die BLObs von gelöschten Dateien werden aus der Datenbank entfernt.
- ♦ **Umbenennen** Die Namen der BLObs werden in der gleichen Weise geändert, wie die Namen der Dateien.

- **Berechtigung des Benutzers**

Die Berechtigungen des Benutzers sind unabhängig von den Berechtigungen unter Windows. Es werden keine unterschiedlichen Berechtigungssysteme implementiert, alle Windows-Benutzer haben alle Zugriffsrechte. Das System lässt grundsätzlich die Vergabe von unterschiedlichen Datei-Berechtigungen zu.

Referenzimplementation - Vorbereitung der Anwendung

Beschreibung der Vorbereitung der Anwendung

Die Referenzimplementation besteht aus fünf CONZEPT 16 Prozeduren und der Konfiguration des Laufwerkstreibers und des SOA-Tasks. Der SOA-Service und der Laufwerkstreiber müssen bereits installiert sein (siehe [Installation](#)).

### Die Prozeduren

Die Prozeduren befinden sich in der Beispieldatenbank "CodeLibrary". Die Datenbank kann aus dem Kunden-Center der vectorsoft AG (siehe [www.vectorsoft.de](http://www.vectorsoft.de)) heruntergeladen werden. Die Implementation besteht aus folgenden Prozeduren:

DrvDef Definition der Ereignisfunktionen

DrvVar Definition von Variablenbereichen und Konstanten

DrvEvt Verarbeitung der Ereignisse des Laufwerkstreibers

DrvSend Aufbereitung der Bestandteile der Datenpakete

DrvMain Aufrufprozedur des SOA-Services

Anpassungen an den Prozeduren müssen nur in der DrvEvt vorgenommen werden.

Nach dem Import der Prozeduren muss in der Funktion DrvEvt:EvtInit (siehe [DrvInit](#)) beim Aufruf von DrvSend:DrvCfgMountLocal() ein freier Laufwerksbuchstabe angegeben werden. Anschließend können alle Prozeduren übersetzt werden. Die Angabe eines temporären Pfades und einer Netzwerkfreigabe ist optional.

### Konfiguration des SOA-Tasks

Zur Konfiguration des SOA-Tasks muss der [SOA-Service](#) angehalten werden. Anschließend können die Konfigurationsdateien angepasst werden (siehe auch [Speicherorte von Konfigurationsdateien](#)).

In der Konfigurationsdatei des SOA-Service (siehe [c16\\_soa.cfg](#)) muss folgender Abschnitt hinzugefügt werden:

```
[DrvSoaSocket]autostart      = Ydescription = DrvSoaSocketmode      = SOCKETstart_delay = 3s
```

Damit das System verwendet werden kann muss hier ebenfalls noch ein Abschnitt für den Laufwerkstreiber eingefügt werden (siehe [Konfiguration des Laufwerkstreibers](#)).

In dem Verzeichnis, in dem sich die Konfigurationsdatei des SOA-Service befindet, muss die Datei DrvSoaSocket.cfg mit folgendem Inhalt erstellt werden:

```
c16_connection_max      = 20c16_connection_shared = Yc16_connection_timeout = 1mc16_database
```

Es müssen noch der symbolische Name der Datenbank, der Benutzer und sein Passwort angegeben werden. Der Benutzer in der Datenbank muss ausführende Berechtigungen für die Prozeduren des Laufwerkstreibers und Lese- und Schreibrechte auf den BLOBs besitzen. In der Installation wird davon ausgegangen, dass der [CONZEPT 16-Server](#) auf dem gleichen Rechner gestartet wurde.

### Konfiguration des Laufwerkstreibers

## Kontakt

Zur Konfiguration des SOA-Tasks muss der SOA-Service angehalten werden. Anschließend können die Konfigurationsdateien angepasst werden (siehe auch Speicherorte von Konfigurationsdateien).

In der Konfigurationsdatei des SOA-Service (siehe c16\_soa.cfg) muss folgender Abschnitt hinzugefügt werden:

```
[Drive]autostart      = Ydescription = Laufwerkstreibermode      = DRIVEstart_delay = 10s
```

Der Laufwerkstreiber muss nach dem SOCKET-Task gestartet werden (start\_delay), da sofort nach dem Start die Initialisierungsnachricht (DrvInit) versendet wird. Diese muss vom Task beantwortet werden.

Damit das System verwendet werden kann, muss hier ebenfalls noch ein Abschnitt für den SOCKET-Task eingefügt werden (siehe Konfiguration des SOA-Tasks).

In dem Verzeichnis, in dem sich die Konfigurationsdatei des SOA-Service befindet, muss die Datei Drive.cfg mit folgendem Inhalt erstellt werden:

```
c16_server      = 127.0.0.1socket_port      = 50001socket_timeout = 1m
```

Der Eintrag socket\_port in der Konfigurationsdatei des Laufwerkstreibers muss dem Eintrag in der Konfigurationsdatei des SOA-Tasks entsprechen. In c16\_server muss der Name oder die IP-Adresse des Servers eingetragen werden, von dem die Lizenz für den Laufwerkstreiber bezogen wird. Wird der Laufwerkstreiber mit der Hot-Standby-Option verwendet, muss in c16\_database der Name der Datenbank eingetragen werden.

### Start des SOA-Service

Nach der Anpassung der c16\_soa.cfg und dem Erstellen der DrvSoaService.cfg und Drive.cfg kann der SOA-Service wieder gestartet werden. Einige Sekunden später steht das entsprechende Laufwerk zur Verfügung. Treten bei der Verarbeitung Fehler auf, werden diese in der Protokolldatei des Laufwerkstreibers (Drive.lgb), der Protokolldatei des SOA-Tasks (DrvSoaService.lgb) oder im Benutzerprotokoll der Datenbank (<Datenbankname>.lgu) geschrieben.

### Referenzimplementation - Arbeitsweise der Implementation

#### Beschreibung der Arbeitsweise der Referenzimplementation

Die Referenzimplementation besteht aus fünf Prozeduren. Die Prozeduren sind in zwei Schichten organisiert. Änderungen dürfen nur an den Prozeduren der oberen Schicht (DrvEvt und DrvDef) erfolgen.

In der Prozedur DrvDef sind Konstanten vorhanden, die für jede Anfrage des Laufwerkstreiber eine Funktion in der Prozedur DrvEvt definieren. Durch Änderungen in den Definitionen können andere Funktionen aufgerufen werden, ohne dass Änderungen in der Prozedur DrvMain notwendig werden. Die Prozedur muss lediglich neu übersetzt werden.

Die Prozedur DrvMain wird durch den SOA-Service aufgerufen, sobald vom Laufwerkstreiber eine Nachricht (Request) eintrifft. Innerhalb der Prozedur wird der Inhalt der Nachricht in globale Variablen übertragen. Anschließend wird auf Grund des eingegangenen Nachrichtentyps die entsprechende Funktion in DrvEvt aufgerufen.

Die Funktionen in DrvEvt schreiben mit Hilfe der Funktionen in DrvSend eine Antwort-Nachricht (Response) auf den Socket. Das Abschließen und Versenden der Nachricht erfolgt dann wieder in der DrvMain nach der Rückkehr aus den Funktionsaufrufen in die Prozedur DrvEvt.

### **Verarbeitung am Beispiel des \_DrvReqInit**

Wird eine Nachricht von Typ \_DrvReqInit empfangen, wird in der DrvMain das gesamte Datenpaket gelesen. In dem Datenpaket befinden sich Informationen über den Laufwerkstreiber, den Computer auf dem der Laufwerkstreiber installiert ist und die Version des installierten Treibers. Die Informationen werden in die entsprechenden globalen Variablen gelesen und der Kopf des Antwort-Pakets geschrieben. Der Keep-alive der Verbindung wird auf 60 Sekunden gesetzt. Anschließend wird die Funktion DrvEvt:EvtInit aufgerufen.

In dieser Funktion wird ein Laufwerksbuchstabe, ein Freigabename, das temporäre Verzeichnis und die Größe des Caches angegeben. Durch Aufrufe der Funktionen DrvSend:DrvCfgMountLocal(), DrvSend:DrvCfgMountNetwork() und DrvSend:DrvCfgCacheDisk() werden die Informationen direkt in das Antwort-Paket geschrieben.

Nach der Rückkehr der Funktion in die DrvMain-Prozedur wird das Resultat der aufgerufenen Funktion geschrieben und das Datenpaket abgeschlossen und versendet. Der Laufwerkstreiber kann mit den Informationen das Laufwerk erstellen und den Cache einrichten. Nach einigen Sekunden steht das Laufwerk anderen Applikationen zur Verfügung.

Nach der Initialisierung werden sofort weitere Anfragen durch den Laufwerkstreiber erzeugt, da das Betriebssystem Informationen über das Laufwerk anfordert. Durch das Einrichten eines Keep-alive bei der Verbindung, entfällt der Verwaltungsaufwand, für diese Anfragen erneut eine Verbindung herzustellen. Daraus resultiert eine erhebliche

Verbesserung der Performanz gegenüber Verbindungen ohne Keep-alive.

### **Verarbeitung am Beispiel des \_DrvReqGetFileEntries**

Bei einer Nachricht von Typ \_DrvReqGetFileEntries wird der Benutzername und die Domäne des am Betriebssystem angemeldeten Benutzers, sowie das Verzeichnis übertragen, dessen Inhalt gelesen werden soll.

Die Antwort gliedert sich in zwei Teile. Zunächst wird überprüft, ob sich Änderungen ergeben haben. Sollte das nicht der Fall sein, wird nur ein kurzes Datenpaket mit den Bereichen \_DrvItemContentChanged und \_DrvItemTimeout übertragen. Der Laufwerkstreiber erstellt das Verzeichnis aus dem Inhalt seines Caches.

Hat sich das Verzeichnis verändert, müssen alle Verzeichniseinträge übermittelt werden. Der Änderungszustand und das Timeout werden übertragen. Anschließend werden für jeden Eintrag der Dateiname, die Attribute, die Dateizeiten (Erstellungsdatum, letzter Zugriff und letzte Änderung) und die Berechtigungen aller Zugriffs-Ids angegeben. Handelt es sich um einen Verzeichniseintrag, wird angegeben, ob das Verzeichnis leer (keine weiteren Unterverzeichnisse oder Dateien enthalten) ist, bei einer Datei wird die Dateigröße angegeben. Die Angabe der Zugriffsberechtigungen kann entfallen, wenn keine Änderungen aufgetreten sind. Die einzelnen Einträge werden durch ein \_DrvResGetFileEntryData voneinander getrennt. Sind alle Einträge in dem Verzeichnis übertragen, wird als Resultat \_rNoRec gesetzt.

## PHP-Schnittstelle

### Beschreibung der CONZEPT 16-PHP-Schnittstelle

Siehe [Blog](#)



Eine Auflistung der Betriebssysteme, auf denen die PHP-Schnittstelle betrieben werden kann, finden Sie in den [Systemvoraussetzungen](#).

Mit der CONZEPT 16-PHP-Schnittstelle kann aus einem PHP-Skript auf eine CONZEPT 16-Datenbank zugegriffen werden. Die Anweisungen für diesen Zugriff sind in einer Erweiterung (Extension) implementiert. Ist diese Erweiterung PHP bekannt gegeben (siehe [PHP-Schnittstelle - Installation](#)), kann die Datei `php_c16_def.php`, die alle Deklarationen beinhaltet, in PHP-Skripte eingebunden und die Befehle zum Zugriff auf CONZEPT 16-Datenbanken verwendet werden.

Der PHP-Skriptinterpreter kann unter anderem von der Internetseite [www.php.net](http://www.php.net) heruntergeladen werden.

Die PHP-Schnittstelle arbeitet im Client-Server-Betrieb in Verbindung mit einem CONZEPT 16-Server und unterstützt das Netzwerkprotokoll TCP/IP. Neben der Include-Datei ist eine entsprechende Import-Bibliothek vorhanden, über deren Funktionen der Zugriff auf den CONZEPT 16-Server und damit auf die Datenbank erfolgt.

## Arbeitsweise

Das Einbinden von Datenbankinhalten erfolgt über den PHP-Skriptinterpreter. Durch die geladene Erweiterung stehen in den PHP-Skripten zusätzliche Befehle zur Kommunikation mit dem CONZEPT 16-Server zur Verfügung.

Damit Inhalte aus einer Datenbank ermittelt werden können muss die Datenbank innerhalb einer Seite geöffnet werden. Anschließend können über die unterschiedlichen Befehle Datensätze, Selektionsmengen usw. verarbeitet werden. Die CONZEPT 16-PHP-Schnittstelle fungiert dabei als Client gegenüber dem CONZEPT 16-Server. Anweisungen innerhalb des PHP-Skripts werden in Anfragen an den Server übersetzt und das Resultat an das PHP-Skript zurückgegeben. Die zurückgegebenen Werte können dann zur Steuerung des Skripts oder zur Darstellung in HTML-Seiten verwendet werden. Am Ende des PHP-Skripts muss die Verbindung zur Datenbank wieder geschlossen werden.

## Funktionsumfang

Der Umfang und die Art der Funktionen ist 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 Datenstrukturinformationen sind alle wesentlichen Funktionen für die Datenbearbeitung (`RecRead()`, `RecLink()`, `RecInsert()`, usw.) und den Zugriff auf Selektionen (`SelOpen()`, `SelRead()`, `SelRun()`, usw.) vorhanden (vgl. [Funktionen der PHP-Schnittstelle](#)).



### PHP-Schnittstelle - Installation

Beschreibung der Installation der CONZEPT 16-PHP-Schnittstelle

Die Dateien der PHP Schnittstelle werden über die CONZEPT 16 Installationsroutine eingerichtet. Die Programmdateien befinden sich anschließend im Installationsverzeichnis unter \Php.

#### *Dateien der PHP-Schnittstelle*

php5.2_c16_x32.dll	PHP-Schnittstelle für Windows-Systeme (PHP 5.2.x)
php5.2ts_c16_x32.dll	PHP-Schnittstelle für Windows-Systeme (PHP 5.2.x, thread-sicher)
php5.2_c16_x32.so	PHP-Schnittstelle für Linux-Systeme (PHP 5.2.x)
php5.2_c16_x64.so	PHP-Schnittstelle für Linux 64 Bit-Systeme (PHP 5.2.x)
php5.3_c16_x32.dll	PHP-Schnittstelle für Windows-Systeme (PHP 5.3.x)
php5.3ts_c16_x32.dll	PHP-Schnittstelle für Windows-Systeme (PHP 5.3.x, thread-sicher)
php5.3_c16_x32.so	PHP-Schnittstelle für Linux-Systeme (PHP 5.3.x)
php5.3_c16_x64.so	PHP-Schnittstelle für Linux 64 Bit-Systeme (PHP 5.3.x)
php5.4_c16_x32.dll	PHP-Schnittstelle für Windows-Systeme (PHP 5.4.x)
php5.4ts_c16_x32.dll	PHP-Schnittstelle für Windows-Systeme (PHP 5.4.x, thread-sicher)
php5.4_c16_x32.so	PHP-Schnittstelle für Linux-Systeme (PHP 5.4.x)
php5.4_c16_x64.so	PHP-Schnittstelle für Linux 64 Bit-Systeme (PHP 5.4.x)
\scripts	Verzeichnis für Beispielskripts
c16.css	Beispiel für ein "Cascading Style-Sheet"
c16_call.php	CONZEPT 16-Funktion aufrufen
c16_catalog.php	Daenstruktur auslesen
c16_info.php	Allgemeine Informationen über den Status des PHP-Interpreters
c16_reclink.php	Verknüpfte Datensätze ausgeben
c16_recop.php	Datensatzverarbeitung
c16_selection.php	Selektionen durchführen
def_connect.php	Verbindungseinstellungen zum Datenbank-Server
def_table.php	Funktionen zum Erstellen von Tabellen
php_c16_def.php	Definitionsdatei für Konstanten




Da derzeit für Windows-Plattformen keine 64-Bit PHP-Interpreter angeboten werden, können dort 32-Bit Interpreter mit der 32-Bit PHP-Schnittstelle von CONZEPT 16 verwendet werden.

### **Thread-Sicherheit**

Neben der PHP-Version, dem Systemtyp und dem Betriebssystem sind PHP-Erweiterungen auch von der Thread-Sicherheit der PHP-Installation abhängig.

Bei der Windows-Version der Web-Umgebung "XAMPP" mit dem Webserver "Apache" wird in der Regel die thread-sichere Variante von PHP verwendet. Der "Internet


Information Services" (IIS) verwendet im Gegensatz dazu üblicherweise die nicht-thread-sichere Variante.

 Je nach verwendeter Umgebung ist unter Windows die thread-sichere oder die nicht-thread-sichere Variante der PHP-Schnittstelle einzusetzen. Die jeweils andere Version kann nicht geladen werden.

Bei der Installation von PHP unter Linux kommt hingegen typischerweise die nicht-thread-sichere Variante zum Einsatz.

## Voraussetzungen

Damit der PHP-Interpreter die CONZEPT 16 PHP-Schnittstelle verwenden kann, muss diese in der Datei php.ini mit Name und Pfad eingetragen oder innerhalb eines PHP-Skripts geladen werden.

 Zum Betrieb der PHP-Schnittstelle muss eine funktionsfähige PHP-Skriptsprache der Version 5.2, 5.3 oder 5.4 installiert sein. Niedrigere oder höhere PHP-Releasestände werden nicht unterstützt.

## Beispiel für den Eintrag in der php.ini

```
extension=php5.4_c16_x32.dll
```

Unter Linux-Systemen wird eine andere Datei eingebunden:

```
extension=php5.4_c16_x32.so
```

oder

```
extension=php5.4_c16_x64.so
```

Stehen die Extensions nicht im gleichen Verzeichnis, wie der Skript-Interpreter, muss der Eintrag "extension\_dir" auf das entsprechende Verzeichnis gesetzt werden.


## Beispiel zum Laden der CONZEPT 16-PHP-Schnittstelle innerhalb des PHP-Skripts

### Anweisungen für Linux-Betriebssystem

```
<?php if (!extension_loaded('CONZEPT 16'))    dl('php5.4_c16_x32.so'); ?> <?php if (!extension_loaded('CONZEPT 16'))    dl('php5.4_c16_x32.so'); ?>
```

Temporäre Dateien, die von der PHP-Schnittstelle angelegt werden, werden unter Windows im Pfad für temporäre Dateien gespeichert (Systemvariablen TEMP oder TMP). Unter Linux werden temporäre Dateien in das Verzeichnis /tmp gespeichert. Soll ein anderes Verzeichnis verwendet werden, muss folgender Eintrag in die Datei php.ini eingetragen werden:

```
[CONZEPT 16]c16.path.temp = <Pfadname>
```

 Nach der Änderung der Datei muss die PHP-Erweiterung neu gestartet werden.

Zur Überprüfung der Installation kann das Programm "php.exe" oder "php-cgi.exe" verwendet werden. Das Programm ist in der Installation von PHP enthalten und stellt den Interpreter der PHP-Skripten dar. Die Installation der PHP-Schnittstelle kann mit

## Kontakt

dem Parameter -i überprüft werden. In der ausgegebenen HTML-Seite werden im Abschnitt "CONZEPT 16 " Informationen zur PHP-Schnittstelle angezeigt.

Im normalen Betrieb erwartet der Interpreter eine Datei mit PHP-Code und generiert daraus zum Beispiel eine HTML-Seite. Die gleiche Ausgabe von php.exe -i erhält man, wenn eine Datei mit folgendem Inhalt übergeben wird:

```
<html><head></head><body><?php  phpinfo();?></body></html>
```

Dies ist eine Datei mit einem PHP-Skript. Wird diese Datei mit dem Namen info.php gespeichert, kann sie mit dem PHP-Interpreter aufgerufen und die Ausgabe in eine Datei mit Namen info.htm umgeleitet werden. Dies kann über folgende Anweisung erfolgen: php info.php > info.htm. Die entstandene HTML-Datei kann in einem Browser angezeigt werden.

Eine erzeugtes PHP-Skript kann über die gleiche Vorgehensweise einfach getestet werden.

## PHP-Schnittstelle - Funktionen

Übersicht über die Funktionen der PHP-Schnittstelle

In der PHP-Schnittstelle sind Funktionen für folgende Bereiche definiert:

- PHP-Datenbank-Funktionen
- PHP-Datensatz-Funktionen
- PHP-Funktionen für Selektionen
- PHP-Funktionen zum Ermitteln von Informationen
- PHP-Satzpuffer-Funktionen
- Sonstige PHP-Funktionen

PHP-Schnittstelle - Datenbankfunktionen

Übersicht über die Datenbankfunktionen der PHP-Schnittstelle

- c16\_close
- c16\_connect

```
c16_connect(aServerName : string,  
aServerPassword : string,  
aAreaName : string, aUserName :  
string, aUserPassword : string[,  
aLocalDsPath : string]) : resource  
Datenbank öffnen
```

aServerName	Protokoll und Name des Servers
aServerPassword	Kennwort des Servers
aAreaName	Symbolischer Name der Datenbank
aUserName	Benutzer
aUserPassword	Kennwort des Benutzers
aLocalDs	Pfad der lokalen Datenstruktur (optional)
Resultat	resource Verbindungsobjekt oder Fehlerwert
Siehe	<u>Befehle der PHP-Schnittstelle</u> , <u>c16_close()</u>

Mit diesem Befehl wird eine Verbindung zum Server aufgebaut, die angegebene Datenbank geöffnet und ihre Datenstruktur in den Hauptspeicher geladen.


Die Anmeldung an den Server geschieht über die Parameter (aServerName) und (aServerPassword). Der Servername setzt sich aus dem verwendeten Protokoll und dem Servernamen zusammen (<Protokoll>:<Servername>). Bei <Protokoll> kann nur "TCP" angegeben werden.

Der Parameter (aServerPassword) beinhaltet eine Zeichenkette mit dem 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 ein Leerstring übergeben.


(aAreaName) enthält eine Zeichenkette mit dem symbolischen Namen der Datenbank. Dies ist der gleiche Name, der auch zur Anmeldung der Datenbank beim Server angegeben wurde. Alternativ kann hier auch der Pfad und der Dateiname der Datenbank (ohne Dateierweiterung .CA1) 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) (Zeichenkette) übergeben. Das Passwort steht in (aUserPassword) (Zeichenkette). Hat der Benutzer kein Passwort, wird eine leere Zeichenkette übergeben. Besitzt der Benutzer eine limitierte Datenstruktur, wird auch nur diese geladen.

 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.

Soll eine lokale Datenstruktur abgelegt werden, muss in dem Parameter (aLocalDs) ein Pfad und ein Dateiname angegeben werden. Die Datenstruktur wird dann lokal abgelegt und muss nicht bei jedem Öffnen der Datenbank übertragen werden. Der Pfad muss mit "/" anstelle von "\" angegeben werden (zum Beispiel c:/c16/dictionary). Die Dateierweiterung .dst wird automatisch an die Datei angehängt und darf nicht mit angegeben werden.

 Bei Verwendung der lokalen Datenstruktur wird keine limitierte Datenstruktur geladen.

Der Rückgabewert ist immer ein Verbindungsobjekt. Ob eine Verbindung zustande gekommen ist, muss mit der Anweisung c16\_error() bestimmt werden. Die Anweisung liefert 0 bzw. C16\_OK oder einen negativen Fehlerwert (siehe Fehlerwerte) zurück.

Es können mehrere Datenbanken gleichzeitig geöffnet sein. Für jede geöffnete Datenbank muss eine eigene Verbindung hergestellt werden.

### Beispiel:

```
$connection = c16_connect('TCP:10.1.0.2+10.1.0.1', '', 'Examples', 'USER', '');if (c16_error($con
```

Folgende Fehler können nach der Ausführung der Anweisung auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_COMM_FAILED (-302)	Kommunikation fehlgeschlagen.
C16ERR_SERVER_OP (-303)	Serverseitige Verarbeitung fehlgeschlagen.
C16ERR_SWAPINIT_FAILED (-305)	Initialisierung der Auslagerungsdatei fehlgeschlagen. Möglicherweise ist in <u>c16.path.temp</u> ein ungültiges Verzeichnis angegeben.
C16ERR_NO_SERVER_CONNECTION (-401)	Serververbindung konnte nicht hergestellt werden.
C16ERR_AREA_NOT_FOUND (-402)	Datenbank nicht gefunden.
C16ERR_AREA_OPEN (-403)	Öffnen der Datenbank fehlgeschlagen.
C16ERR_AREA_LOCKED (-404)	Datenbank gesperrt.
C16ERR_AREA_IN_USE (-405)	Datenbank in exklusiver Benutzung.
C16ERR_AREA_TYPE (-406)	Datenbank nicht kompatibel.
C16ERR_AREA_PASSWORD (-407)	Serverpasswort nicht korrekt.
C16ERR_SERVER_USER_LIMIT (-408)	Benutzerlimit des Servers erreicht.
C16ERR_SERVER_START (-409)	Serverprozess konnte nicht gestartet werden.

## Kontakt

C16ERR_USER_INVALID (-413)	Benutzername oder -passwort nicht korrekt.
C16ERR_AREA_STANDBY (-416)	Datenbank im Standby-Modus.
C16ERR_AREA_ROLLBACK (-417)	Datenbank im Rollback-Modus.
C16ERR_AREA_LOCKED_ADMIN (-418)	Datenbank durch Administrator gesperrt.
C16ERR_AREA_LOCKED_OPERATION (-419)	Datenbank durch Serviceoperation gesperrt.
C16ERR_AREA_LOCKED_DOWN (-420)	Datenbank wird geschlossen.
C16ERR_AREA_LOCKED_STANDBY (-421)	Datenbank durch Standby-Modus gesperrt.
C16ERR_AREA_LOCKED_ROLLBACK (-422)	Datenbank durch Rollback-Sperre gesperrt.
C16ERR_AREA_LOCKED_OPEN (-423)	Datenbank durch Login-Sperre gesperrt.
C16ERR_AREA_LOCKED_NO_STANDBY_OPEN (-424)	Datenbank durch Standby-Sperre gesperrt.
C16ERR_LICENSE_SRVNOTSUPPORTED (-2001)	Version des Datenbankservers nicht unterstützt.
C16ERR_LICENSE_CLNUPGRADE (-2002)	Version des Clients nicht lizenziert.
C16ERR_LICENSE_CLNNOTSUPPORTED (-2003)	Typ des Clients nicht lizenziert.
C16ERR_LICENSE_COMM (-2004)	Kommunikation mit Lizenzserver fehlgeschlagen.
C16ERR_LICENSE_DATA (-2005)	Datenaustausch mit Lizenzserver fehlgeschlagen.
C16ERR_LICENSE_INVALID (-2006)	Lizenzdaten ungültig.
C16ERR_LICENSE_MISMATCH_EVAL (-2007)	Client nicht von Evaluierungslizenz unterstützt.



c16\_close(aConnection :  
resource) : int  
Datenbank schließen  
aConnection Verbindungsobjekt  
Resultat int Fehlerwert  
Befehle der  
Siehe PHP-Schnittstelle,  
c16\_connect()

Diese Funktion schließt eine mit c16\_connect() geöffnete Datenbank.

In dem Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt angegeben.

Der Rückgabewert vom Typ int beinhaltet entweder C16\_Ok (kein Fehler) oder einen Fehlerwert.



Diese Funktion muss vor Programmende für jede geöffnete Datenbank aufgerufen werden.

**Beispiel:**

```
$connection = c16_connect('TCP:10.1.0.2+10.1.0.1', '', 'Examples', 'USER', '');if (c16_error($con
```

Von der Anweisung können folgende Resultatwerte zurückgegeben werden:

C16_OK (0)	Kein Fehler aufgetreten.
C16_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16_ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.

## PHP-Schnittstelle - Datensatzfunktionen

Übersicht über die Datensatzfunktionen der PHP-Schnittstelle

- [c16\\_recdelete](#)
- [c16\\_recdeleteall](#)
- [c16\\_recinsert](#)
- [c16\\_reclink](#)
- [c16\\_recread](#)
- [c16\\_recreplace](#)

c16\_recdelete(aConnection :  
 resource, aFileNo : int, aOptions :  
 int) : int  
 Datensatz löschen  
 aConnection Verbindungsobjekt  
 aFileNo Dateinummer  
 Optionen  
 0 Aktueller  
 Datensatz  
 (Feldpuffer  
 des ersten  
 Schlüssels)  
\_RecFirst Erster  
 Datensatz  
 aOptions \_RecPrev Vorheriger  
 Datensatz  
\_RecNext Nächster  
 Datensatz  
\_RecLast Letzter  
 Datensatz  
\_RecUseID Datensatz-ID  
 zur  
 Löschung  
 verwenden  
 Resultat int Fehlerwert  
Befehle der  
 Siehe PHP-Schnittstelle,  
c16\_recinsert(),  
c16\_recdeleteall()

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 (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Der Parameter (aOptions) bestimmt, welcher Datensatz gelöscht wird.

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

## Kontakt

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.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.

c16\_recdeleteall(aConnection :  
resource, aFileNo : int) : int  
Alle Datensätze einer Datei löschen  
aConnection Verbindungsobjekt

aFileNo      Dateinummer  
Resultat     int Fehlerwert

Siehe         Befehle der  
                PHP-Schnittstelle,  
                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 (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.

c16\_recinsert(aConnection : resource,  
aFileNo : int, aOptions : int) : int  
Datensatz einfügen  
aConnection Verbindungsobjekt

aFileNo	Dateinummer	
	Optionen	
	<u>_RecLock</u>	Datensatz sperren
aOptions	<u>_RecSingleLock</u>	Datensatz einfach sperren
	<u>_RecSharedLock</u>	Datensatz mehrfach sperren

Resultat int Fehlerwert  
Befehle der  
Siehe PHP-Schnittstelle,  
c16\_recdelete(),  
c16\_recreplace()

Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt 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 (aOptions) 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	<u>_rOk</u>	Datensatzoperation erfolgreich.
6	<u>_rExists</u>	Der Datensatz konnte nicht eingefügt oder zurückgespeichert werden, da ein Satz mit einem identischen eindeutigen Schlüsselwert bereits existiert.
10	<u>_rDeadlock</u>	Der Datensatz konnte aufgrund einer Verklemmung nicht eingefügt werden.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.

c16\_recread(aConnection :  
resource, aFileNo : int,  
aKeyNoOrSel : mixed, aOptions  
: int[, aAddInfo : int]) : int  
Datensatz lesen  
aConnection Verbindungsobjekt  
aFileNo Dateinummer  
Schlüsselnummer  
aKeyNoOrSel oder  
Selektionsobjekt  
aOptions Zugriffsposition  
Zusätzliche  
aAddInfo Information  
(optional)  
Resultat int Fehlerwert  
Siehe Befehle der  
PHP-Schnittstelle,  
c16\_reclink()

Liest einen Datensatz und überträgt ihn in die Feldpuffer.


Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Die Funktion führt einen Zugriff über den Schlüssel (aKeyNoOrSel) in die Datei (aFileNo) durch. Der Schlüssel wird aus den aktuellen Feldinhalten der Datei gebildet. Ist kein Datensatz mit dem entsprechenden Schlüssel vorhanden, 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.

Das Übertragen der Feldpuffer in PHP-Variablen und umgekehrt erfolgt über die Anweisungen c16\_fldget() bzw. c16\_fldset().

Soll auf eine Selektionsmenge zugegriffen werden, wird in dem Parameter (aKeyNoOrSel) das Selektionsobjekt übergeben. Das Selektionsobjekt muss zuvor mit c16\_selopen() angelegt worden sein.

Die Zugriffsposition wird über den Parameter (aOptions) angegeben. Folgende symbolische Konstanten können angegeben werden:

<u>_RecFirst</u>	Der Satz mit dem kleinsten Schlüsselwert wird gelesen.
<u>_RecLast</u>	Der Satz mit dem größten Schlüsselwert wird gelesen.
<u>_RecPrev</u>	Der Satz mit dem nächst kleineren Schlüsselwert wird geladen. Sofern kein weiterer Satz vorhanden ist, wird als Resultat <u>_rNoRec</u> zurückgeliefert.
<u>_RecNext</u>	Der Satz mit dem nächst größeren Schlüsselwert wird geladen. Sofern kein weiterer Satz vorhanden ist, wird als Resultat <u>_rNoRec</u> zurückgeliefert.
<u>_RecPos</u>	Anstatt über den aktuellen Schlüsselwert wird über die Schlüsselposition zugegriffen. Die Position wird in (aAddInfo) übergeben.

- `_RecLock` Der gelesene Satz wird gesperrt. Dies ist nur dann der Fall, wenn `_rOk` zurückgeliefert wird (eindeutiger Schlüssel). Ist der Satz durch einen anderen Benutzer gesperrt, so ist das Resultat `_rLocked`.
- `_RecForceLock` Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn `_rOk` zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, 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.
- `_RecUnlock` Der gelesene Satz wird entsperrt.
- `_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.
- `_RecID` Es wird der Datensatz mit einer bestimmten Datensatz-ID gelesen. Der Parameter (`aAddInfo`) ist optional, er hat zwei verschiedene Bedeutungen. Bei der Benutzung der Option `_RecPos` wird hier die gewünschte Schlüsselposition übergeben. Ist der Parameter (`aKeyNoOrSel`) gleich 0, 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, wird `_rNoRec` zurückgeliefert und kein Satz geladen. Dies gilt nicht, wenn `_RecFirst`, `_RecLast`, `_RecNext` oder `_RecPrev` benutzt 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.
- 2 `_rMultiKey` Der Schlüssel ist nicht eindeutig. In der Datei sind mehrere Sätze mit dem gewünschten Schlüsselwert vorhanden, der erste Satz wurde geladen.
- 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`



## Kontakt

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.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aKeyNoOrSel übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_KEY (-604)	Schlüssel nicht vorhanden.

c16\_recreplace(aConnection :  
resource, aFileNo : int, aOptions :  
int) : int

Datensatz ersetzen

aConnection Verbindungsobjekt

aFileNo Dateinummer

Optionen

0

Datensatzsperre

aOptions beibehalten

RecUnlock

Datensatz

entsperren

Resultat int Fehlerwert

Befehle der

PHP-Schnittstelle,

Siehe

c16\_recdelete(),

c16\_recinsert()

Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt 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 (aOptions) kann angegeben werden, ob der Datensatz nach dem Rückspeichern weiter gesperrt bleiben soll oder nicht. Wird keine der beiden Optionen benutzt, 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.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

- C16\_OK (0) Kein Fehler aufgetreten.
- C16ERR\_ARGS\_FORMAT (-251) Anzahl der Argumente oder Typ eines Arguments ungültig.
- C16ERR\_ARGS\_CONNECTION (-254) Kein Verbindungsobjekt in aConnection übergeben.
- C16ERR\_NO\_FILE (-601) Die übergebene Dateinummer existiert nicht.

`c16_reclink(aConnection :`  
`resource, aSrcFileNo : int,`  
`aDstFileNo : int, aLinkNoOrSel :`  
`int, aOptions : int[, aReserved :`  
`int[, aLinkPos : int]) : int`  
 Verknüpften Datensatz lesen  
`aConnection` Verbindungsobjekt  
`aSrcFileNo` Nummer der  
 Ausgangsdatei  
`aDstFileNo` Nummer der  
 Zielfeld  
`aLinkNoOrSel` Nummer der  
 Verknüpfung  
 oder  
 Selektionsobjekt  
`aOptions` Zugriffsposition  
 Reserviert, muss  
`aReserved` 0 enthalten  
 (optional)  
`aLinkPos` Zusätzliche  
 Information zur  
 Zugriffsposition  
 (optional)  
 Resultat `int` Fehlerwert  
 Siehe Befehle der  
PHP-Schnittstelle,  
`c16_recread()`,  
`c16_fldset()`,  
`c16_fldget()`

Der Befehl liest einen verknüpften Datensatz und überträgt ihn in die Feldpuffer. Der Feldpuffer kann anschließend mit der Anweisung `c16_fldget()` in eine PHP-Variable gelesen werden.


Im Parameter (`aConnection`) wird das von `c16_connect()` bereitgestellte Verbindungsobjekt übergeben. Die Funktion führt einen Zugriff auf die Datei (`aDstFileNo`) durch. Aus dem zuvor gelesenen Datensatz aus der Datei (`aSrcFileNo`) werden aus den Verknüpfungsfeldern (inkl. der Felder für Zugriffspositionierung) der Verknüpfung (`aLinkNoOrSel`) ein Schlüsselwert gebildet und auf den Satz in der Zielfeld zugriffen. Ist kein Datensatz mit dem entsprechenden Schlüssel vorhanden, wird der Datensatz mit dem nächst größeren bzw. dem größten Schlüssel geladen.

Soll auf eine Selektionsmenge zugriffen werden, wird in dem Parameter (`aLinkNoOrSel`) das Selektionsobjekt übergeben. Das Selektionsobjekt muss zuvor mit `c16_selopen()` angelegt worden sein.

`_RecFirst` Der erste verknüpfte Satz wird geladen.  
`_RecLast` Der letzte verknüpfte Satz wird geladen.  
`_RecPrev`

## Kontakt

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.
- `_RecPos` Der Zugriff findet über die Verknüfungsposition statt (vgl. `C16_RecLinkInfo()`), 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.
- `_RecLock` 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`.
- `_RecForceLock` Der gelesene Satz wird gesperrt, ohne Rücksicht auf eine Sperre durch einen anderen Benutzer. Die Sperre ist nur dann erfolgt, wenn `_rOk` zurückgeliefert wird. Wenn ein anderer Benutzer den Satz gesperrt haben sollte, 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.
- `_RecUnlock` Der gelesene Satz wird entsperrt.
- `_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 der Zieldatei. 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. Wird in `(aOptions)` die Option `_RecPos` angegeben, muss in `(aLinkPos)` die Position des zu lesenden Datensatzes übergeben werden. Der Parameter `(aReserved)` wird nicht benutzt, er muss 0 enthalten.

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` Es wurde kein Satz geladen, da entweder die Datei leer ist, oder kein vorhergehender bzw. nachfolgender Satz existiert.

## Kontakt

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_SEL_HDL_INVALID (-208)	Selektionsobjekt aLinkNoOrSel ungültig. Möglicherweise wurde die Selektion noch nicht durchgeführt.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aLinkNoOrSel übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_LINK (-606)	Verknüpfung nicht vorhanden.
C16ERR_LINK_INVALID (-608)	Verknüpfung aLinkNoOrSel ungültig.

## PHP-Schnittstelle - Funktionen für Selektionen

Übersicht über die Selektions-Funktionen der PHP-Schnittstelle

- [c16\\_selclear](#)
- [c16\\_selclose](#)
- [c16\\_selcopy](#)
- [c16\\_seldelete](#)
- [c16\\_selignore](#)
- [c16\\_selinfo](#)
- [c16\\_selopen](#)
- [c16\\_selread](#)
- [c16\\_selrecdelete](#)
- [c16\\_selrecinsert](#)
- [c16\\_selrun](#)

c16\_selopen(aConnection :  
resource) : int  
Selektionspuffer anlegen  
aConnection Verbindungsobjekt  
Resultat     int     Selektionsobjekt  
                         oder Fehlerwert  
Siehe           Befehle der  
                 PHP-Schnittstelle,  
                 c16\_selclose(),  
                 c16\_selread()

Diese Funktion legt einen neuen Selektionspuffer an, über den auf Selektionen und deren Mengen zugegriffen werden kann.

Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben.

Der erzeugte Deskriptor wird als Resultat zurückgegeben. Ist der Rückgabewert kleiner oder gleich 0 ist ein Fehler aufgetreten. Die Fehlerwerte sind im Abschnitt Fehlerwerte beschrieben.

### Beispiel:

```
$sel = c16_selopen($connection);if ($sel > 0){ // Selektion lesen $res = c16_selread($sel, 1, _
```

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.

c16\_selread(aSelection :  
 resource, aFileNo : int,  
 aOptions : int[, aName :  
 string]) : int  
 Selektion in den Puffer laden  
 aSelection Selektionsobjekt  
 aFileNo Dateinummer  
 aOptions Optionen  
 aName Name (optional)  
 Resultat int Fehlerwert  
 Siehe [Befehle der  
 PHP-Schnittstelle,  
 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 (aSelection) wird das von [c16\\_selopen\(\)](#) bereitgestellte Selektionsobjekt übergeben.

In (aFileNo) wird die Nummer der Datei angegeben, in der die Selektion definiert ist. Der Name der Selektion in (aName) kann entfallen, wenn \_SelFirst, \_SelLast, \_SelPrev oder \_SelNext in (aOptions) angegeben wird.

In (aOptions) 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 exklusiv gesperrt. Dies ist nur dann der Fall, wenn _rOk zurückgeliefert wird (Selektion vorhanden). Ist die Selektion durch einen anderen Benutzer gesperrt, ist das Resultat _rLocked.
_SelSharedLock	Die gelesene Selektion wird nicht exklusiv gesperrt. Andere Benutzer können noch lesend auf die Selektion zugreifen und die Selektion mit der gleichen Option zusätzlich sperren. Mit dieser Sperr-Option kann die Selektion nur gelesen, nicht aber geändert werden. Soll die Selektion verändert werden, muss sie mit der Option _SelLock gesperrt werden.
_SelUnlock	Die gelesene Selektion wird entsperrt.
_SelKeyMode	Alternativer Verarbeitungsmodus
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 das Selektionsobjekt angegeben wird.

Der Erfolg der Operation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- 0 \_rOK        Selektion gelesen.
- 1 \_rLocked   Selektion ist vorhanden und von einem anderen Benutzer gesperrt.
- 3 \_rNoKey    In der Datei ist keine Selektion mit dem gewünschten Namen vorhanden.  
              Es wurde die nächste Selektion gelesen.
- 4 \_rLastRec   Selektion nicht vorhanden, letzte Selektion gelesen.
- 5 \_rNoRec    Keine weitere Selektion vorhanden.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aSelection übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.

c16\_selrun(aSelection : resource,  
aOptions : int[, aName : string]) : int  
Selektion durchführen  
aSelection Selektionsobjekt

Optionen:

\_SelBase Nachselektion

\_SelUnion Vereinigungsmenge

aOptions bilden

\_SelInter Schnittmenge  
bilden

\_SelMinus Restmenge bilden

aName Name der zweiten Selektion  
(optional)

Resultat int Fehlerwert

Befehle der

Siehe PHP-Schnittstelle,  
c16\_selopen(), c16\_selread()

Diese Funktion führt die Selektion (aSelection) durch. Der Puffer muss zuvor mit c16\_selopen() angelegt und eine Selektion mit c16\_selread(..., \_SelLock,...) gelesen und gesperrt werden. Es ist zu beachten, dass c16\_selrun() die Feldpuffer der Selektionsdatei verändert. Bei Angabe von \_SelBase, \_SelUnion, \_SelInter und \_SelMinus muss eine zweite Selektion in (aName) angegeben werden.

Über die Funktion c16\_recread() kann auf die Selektionsmenge zugegriffen werden.

Die Anzahl der selektierten Datensätze kann über die Funktion c16\_recinfo() und bei einer Wertemengenselektion über die Funktion c16\_selinfo() ermittelt werden.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aSelection übergeben.

c16\_selignore(aSelection :  
resource, aKeyNo : int, aIgnore :  
bool) : int

Selektionsvorauswahl ignorieren

aSelection Selektionsobjekt

aKeyNo Schlüsselnummer

Vorauswahl

aIgnore ignorieren (1),  
nicht ignorieren  
(0)

Resultat int Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle

Mit diesem Befehl können beim nächsten c16\_selrun() Vorauswahlen der Selektion ignoriert werden. In (aSelection) wird das von c16\_selopen() bereitgestellte Selektionsobjekt übergeben. aKeyNo gibt die Nummer des Schlüssels an, bei dem keine Vorauswahlen durchgeführt werden sollen. Dadurch kann in einer Prozedur die Anzahl durchzuführender Vorauswahlen reduziert werden. Dies ist insbesondere mit dem Selektions-Parameter "Nur Vorauswahl" sinnvoll. Nach dem nächsten c16\_selrun() sind alle durch c16\_selignore() vorgenommenen Einschränkungen wieder aufgehoben.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aSelection übergeben.

c16\_selclose(aSelection :  
resource) : int  
Selektionspuffer entfernen  
aSelection Selektionsobjekt  
Resultat int Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle,  
c16\_selopen()

Mit dieser Funktion wird der durch c16\_selopen() erzeugte Selektionspuffer wieder entfernt.

Im Parameter (aSelection) wird das von c16\_selopen() bereitgestellte Selektionsobjekt übergeben.

Der Rückgabewert beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

### **Beispiel:**

```
$sel = c16_selopen($connection);if ($sel > 0){ // Selektion lesen $res = c16_selread($sel, 1, _
```

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aSelection übergeben.

`c16_selclear(aSelection : resource) :`  
`int`

Inhalt des Selektionspuffers löschen

`aSelection` Selektionsobjekt

Resultat `int` Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle,  
`c16_selrecdelete()`,  
`c16_selrecinsert()`

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 Ergebnismenge. Dadurch kann beispielsweise die Ergebnismenge mit `c16_selrecinsert()` gefüllt werden.

Im Parameter (`aSelection`) wird das von `c16_selopen()` bereitgestellte Selektionsobjekt übergeben.

Der Rückgabewert vom Typ `int` beinhaltet entweder 0 (kein Fehler) oder einen negativen Fehlerwert (siehe Fehlerwerte).

0 `_rOK` Datensatzoperation erfolgreich.

7 `_rNoLock` Die Selektion ist nicht gesperrt.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen `c16_error()` oder `c16_errortext()` ermittelt werden. Folgende Fehler können auftreten:

<code>C16_OK (0)</code>	Kein Fehler aufgetreten.
<code>C16ERR_ARGS_FORMAT (-251)</code>	Anzahl der Argumente oder Typ eines Arguments ungültig.
<code>C16ERR_ARGS_SELECTION (-255)</code>	Kein Selektionsobjekt in <code>aSelection</code> übergeben.

c16\_selcopy(aConnection :  
 resource, aFileNo : int,  
 aNameSrc : string, aNameDst :  
 string) : int  
 Selektion kopieren  
 aConnection Verbindungsobjekt  
 aFileNo Dateinummer  
 aNameSrc Name der  
 Quell-Selektion  
 aNameDst Name der  
 Ziel-Selektion  
 Resultat int Fehlerwert  
Befehle der  
 Siehe PHP-Schnittstelle,  
c16\_seldelete()

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

In (aConnection) wird das von dem Befehl c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Der Parameter (aFileNo) enthält die Dateinummer, in der die zu kopierende Selektion definiert ist. Die zu kopierende Selektion wird in Parameter (aNameSrc) angegeben, in Parameter (aNameDst) wird der Name der durch Kopieren neu erstellten Selektion bestimmt.

c16\_selcopy() ermöglicht es, in einer Mehrbenutzerumgebung mehrere Benutzer gleichzeitig identische Selektionen durchführen zu lassen. Dazu wird eine Master-Selektion erstellt, die dann für den jeweiligen Benutzer temporär unter einem anderen Namen kopiert wird.

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



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

### Beispiel:

```
// Name der Selektion erzeugen$selnew = 'TMP_SEL_' . session_name();// Selektion kopieren$res = c16_selcopy($aConnection, $aFileNo, $aNameSrc, $aNameDst);
```

Der Erfolg der Datensatzoperation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

- |   |                       |   |
|---|-----------------------|---|
| 0 | <code>_rOK</code>     | Datensatzoperation erfolgreich.   |
| 1 | <code>_rLocked</code> | Datensatz ist vorhanden und von einem anderen Benutzer gesperrt. Der Satz wurde geladen, sofern die Option <code>_RecNoLoad</code> nicht angegeben wurde. |
| 3 | <code>_rNoKey</code>  | In der Datei ist kein Satz mit dem gewünschten Schlüsselwert vorhanden. Es wurde der Satz mit dem nächst größeren Schlüsselwert                           |

geladen.

5 \_rNoRec Es wurde kein Satz geladen, da entweder die Datei leer ist, oder kein vorhergehender bzw. nachfolgender Satz existiert.

6 \_rExists Die Zielselektion existiert bereits.

10 \_rDeadlock Es ist eine Verklemmung aufgetreten.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16\_OK (0)

Kein Fehler aufgetreten.

C16ERR\_ARGS\_FORMAT (-251)

Anzahl der Argumente oder Typ eines Arguments ungültig.

C16ERR\_ARGS\_VALUE (-253)

Wert eines Arguments ungültig.

C16ERR\_ARGS\_CONNECTION  
(-254)

Kein Verbindungsobjekt in aConnection übergeben.

c16\_seldelete(aConnection :  
resource, aFileNo : int[, aName  
: string]) : int

Selektion löschen

aConnection Verbindungsobjekt

aFileNo Dateinummer

aName Name (optional)

Resultat int Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle,  
c16\_selcopy()

Mit dieser Funktion kann eine bestehende Selektion komplett entfernt werden, einschließlich eventuell vorhandener Ergebnismengen. Dabei wird in Parameter (aConnection) das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Die Nummer der Datei und der Name der zu löschenden Selektion wird in (aFileNo) und (aName) angegeben.

Über den Rückgabewert kann geprüft werden, ob die Selektionsmenge gelöscht werden konnte.

Der Erfolg der Operation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

0 \_rOK Datensatzoperation erfolgreich.

1 \_rLocked Selektion ist vorhanden und von einem anderen Benutzer gesperrt.

5 \_rNoRec Die Selektion wurde nicht gefunden.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.



c16\_selrecdelete(aSelection : resource,  
aFileNo : int) : int

Datensatz aus der Selektion entfernen

aSelection Selektionsobjekt

aFileNo Dateinummer

Resultat int Fehlerwert

Befehle der  
PHP-Schnittstelle,  
Siehe c16\_selread(),  
c16\_selclear(),  
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 dem tatsächlichen Sortierungswert im Datensatz übereinstimmen muss, da sonst der Datensatz nicht gelöscht werden kann (beispielweise ist die Selektionsmenge nach Namen sortiert, der Name im Datensatz hat sich mittlerweile aber verändert). 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 (aSelection) wird das von c16\_selopen() bereitgestellte Selektionsobjekt übergeben.

Der Erfolg der Operation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

0 \_rOK Datensatzoperation erfolgreich.

3 \_rNoKey In der Selektion ist kein Satz mit dem gewünschten Schlüsselwert vorhanden.

5 \_rNoRec In der Selektion ist keine Datensatz vorhanden.

6 \_rNoLock Die Selektion ist nicht gesperrt.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16\_OK (0) Kein Fehler aufgetreten.

C16ERR\_ARGS\_FORMAT (-251) Anzahl der Argumente oder Typ eines Arguments ungültig.

C16ERR\_ARGS\_SELECTION (-255) Kein Selektionsobjekt in aSelection übergeben.

C16ERR\_NO\_FILE (-601) Datei nicht vorhanden.

c16\_selrecinsert(aSelection :  
resource, aFileNo : int) : int  
Datensatz der Selektion hinzufügen  
aSelection Selektionsobjekt  
aFileNo Dateinummer  
Resultat int Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle,  
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 (aSelection) wird das von c16\_selopen() bereitgestellte Selektionsobjekt übergeben.

Der Erfolg der Operation wird von der Funktion zurückgegeben. Bei der Überprüfung können folgende symbolische Konstanten verwendet werden:

0 \_rOK            Datensatzoperation erfolgreich.  
3 \_rExists       Der Datensatz ist bereits in der Selektion vorhanden.  
6 \_rNoLock      Die Selektion ist nicht gesperrt.  
5 \_rDeadlock    Es ist eine Verklemmung aufgetreten.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aSelection übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.

PHP-Schnittstelle - Funktionen zum Ermitteln von Informationen

Übersicht über die Funktionen der PHP-Schnittstelle zum Ermitteln von Informationen aus der Datenstruktur

- [c16\\_fileinfo](#)
- [c16\\_fldinfo](#)
- [c16\\_keyfldinfo](#)
- [c16\\_keyinfo](#)
- [c16\\_linkfldinfo](#)
- [c16\\_linkinfo](#)
- [c16\\_recinfo](#)
- [c16\\_reclinkinfo](#)
- [c16\\_sbrinfo](#)
- [c16\\_selinfo](#)

`c16_fileinfo(aConnection : resource[, aFileNo : int]) :`

array

Informationen zu einer Datei oder alle Dateien ermitteln

aConnection Verbindungsobjekt

aFileNo Dateinummer (optional)

Resultat array Informationsarray oder Fehlerwert

Siehe Befehle der PHP-Schnittstelle,  
c16\_sbrinfo(), c16\_fldinfo()

Der Befehl gibt ein Array mit Informationen zurück. In (aConnection) wird das von der Anweisung c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Wird (aFileNo) nicht angegeben, wird ein indizierter Array mit den Nummern als Schlüsselwerte und den Dateinamen als Werte zurückgegeben.

Wird in (aFileNo) eine gültige Dateinummer angegeben, wird ein assoziatives Array mit folgenden Schlüsseln zurückgegeben:

### **Schlüssel      Wert**

<code>_FileNumber</code>	Dateinummer
<code>_FileName</code>	Dateiname
<code>_FileMaster</code>	Dateinummer der übergeordneten Datei
<code>_FileSbrCount</code>	Anzahl der Teildatensätze
<code>_FileKeyCount</code>	Anzahl der Schlüssel
<code>_FileLnkCount</code>	Anzahl der Verknüpfungen
<code>_FileUserLevel</code>	Benutzerberechtigung
<code>_FileOemMark</code>	Markierung für das OEM-Kit

### **Beispiele:**

```
// Alle Dateien ermitteln und ausgeben
echo '<table>';
$info = c16_fileinfo($connection);
foreach ($info as $key => $value) {
```

Im Falle eines Fehlers wird der Fehlerwert zurückgegeben. Der Fehlertext kann anschließend mit c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

<code>C16_OK (0)</code>	Kein Fehler aufgetreten.
<code>C16ERR_ARGS_FORMAT (-251)</code>	Anzahl der Argumente oder Typ eines Arguments ungültig.
<code>C16ERR_ARGS_TYPE (-252)</code>	Typ eines mehrtypigen Arguments ungültig.
<code>C16ERR_ARGS_CONNECTION (-254)</code>	Kein Verbindungsobjekt in aConnection übergeben.
<code>C16ERR_RESULT_ARRAY (-256)</code>	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
<code>C16ERR_NO_FILE (-601)</code>	Datei nicht vorhanden.

`c16_sbrinfo(aConnection : resource, aFileNo :  
int[, aSbrNo : int]) : array`  
Informationen zu einem Teildatensatz ermitteln  
aConnection Verbindungsobjekt

aFileNo      Dateinummer  
aSbrNo      Teildatensatznummer (optional)  
Resultat      Array Informationsarray oder  
Fehlerwert

Siehe      Befehle der PHP-Schnittstelle,  
c16\_fileinfo(), c16\_fldinfo()

Der Befehl gibt ein Array mit Informationen zurück. In (aConnection) wird das von der Anweisung `c16_connect()` bereitgestellte Verbindungsobjekt übergeben. In (aFileNo) wird die Dateinummer angegeben. Wird (aSbrNo) nicht angegeben, wird ein indizierter Array mit den Nummern als Schlüssel und den Teildatensatznamen als Werte zurückgegeben.

Wird in (aSbrNo) eine gültige Teildatensatznummer angegeben, wird ein assoziatives Array mit folgenden Schlüsseln zurückgegeben:

<b>Schlüssel</b>	<b>Wert</b>
<code>_FileNumber</code>	Dateinummer
<code>_SbrNumber</code>	Teildatensatznummer
<code>_SbrName</code>	Teildatensatzname
<code>_SbrStatus</code>	Aktivität des Teildatensatzes
<code>_SbrFldCount</code>	Anzahl der Felder

Wird eine nicht vorhandene Teildatensatznummer übergeben, ist das Array leer.

### Beispiele:

```
// Alle Teildatensätze ermitteln und ausgebenecho '<table>';$info = c16_sbrinfo($connection,1);fo
```

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen `c16_error()` oder `c16_errortext()` ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_SBR (-602)	Teildatensatz nicht vorhanden.

c16\_fldinfo(aConnection : resource,  
aFileNo : int, aSbrNo : int[, aFld :  
mixed]) : array

Informationen über ein Feld ermitteln

aConnection Verbindungsobjekt

aFileNo Dateinummer

aSbrNo Teildatensatznummer

aFld Feldnummer oder  
Feldname (optional)

Resultat array Informationsarray  
oder Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle,  
c16\_fileinfo(),  
c16\_sbrinfo()

Der Befehl gibt ein Array mit Informationen zurück. In (aConnection) wird das von der Anweisung c16\_connect() bereitgestellte Verbindungsobjekt übergeben. In (aFileNo) und (aSbrNo) wird die Nummer der Datei und des Teildatensatzes übergeben. Wird (aFld) nicht angegeben, wird ein indizierter Array mit den Nummern als Schlüsselwerte und den Feldnamen als Werte zurückgegeben.

Wird in (aFld) eine gültige Feldnummer oder der Name eines Feldes angegeben, wird ein assoziatives Array mit folgenden Schlüsseln zurückgegeben:

Schlüssel	Wert
_FileNumber	Dateinummer
_SbrNumber	Teildatensatznummer
_FldNumber	Feldnummer
_FldName	Feldname
_FldType	Typ des Feldes
_FldMaxLen	Maximale Länge
_FldInputRight	Eingabeberechtigung
_FldOutputRight	Ausgabeberechtigung

Bei der Angabe des Feldnamens werden die Parameter (aFileNo) und (aSbrNo) ignoriert.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert zurückgegeben. Der Fehler kann ebenfalls über die Anweisungen c16\_error() oder c16\_errortext() ermittelt werden.

Der zurückgegebene Feldtyp kann mit folgenden Konstanten verglichen werden:

#### Wert Konstante

- 1    \_TypeAlpha
- 2    \_TypeDate
- 3    \_TypeByte
- 4    \_TypeWord

- 5    \_TypeDecimal
- 6    \_TypeMemo
- 7    \_TypeInt
- 8    \_TypeBigInt
- 9    \_TypeFloat
- 10   \_TypeLogic
- 11   \_TypeTime

**Beispiele:**

```
// Alle Felder ermitteln und ausgeben
echo '<table>'; $info = c16_fldinfo($connection,1,1); foreach
```

Tritt ein Fehler bei der Verarbeitung auf, kann der zurückgegebene Fehlerwert mit folgenden Konstanten verglichen werden:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_SBR (-602)	Teildatensatz nicht vorhanden.
C16ERR_NO_FLD (-603)	Feld nicht vorhanden.

`c16_keyinfo(aConnection : resource,  
aFileNo : int[, aKey : mixed]) : array`  
Informationen über Schlüssel ermitteln  
aConnection Verbindungsobjekt

aFileNo Dateinummer  
aKey Schlüsselnummer oder  
Schlüsselname (optional)

Resultat mixed Informationsarray  
oder Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle,  
c16\_keyfldinfo()

Der Befehl gibt ein Array von Informationen über alle oder einen Schlüssel zurück.

Im Parameter (aConnection) wird das von `c16_connect()` bereitgestellte Verbindungsobjekt übergeben. In den Parametern (aFileNo) muss die Dateinummer übergeben werden. Wird (aKey) nicht übergeben, wird ein indizierter Array mit den Schlüsselnummern als Index und den Schlüsselnamen als Wert zurückgegeben. Wird eine gültige Schlüsselnummer übergeben, wird ein assoziatives Array mit folgenden Informationen zurückgegeben:

\_FileNumber Nummer der Datei  
\_KeyNumber Nummer des Schlüssels  
\_KeyName Name des Schlüssels  
\_KeyFldCount Anzahl der Schlüsselfelder  
\_KeyIsUnique Ein- oder mehrdeutiger Schlüssel

Bei einem Fehler wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen `c16_error()` oder `c16_errortext()` ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_KEY (-604)	Schlüssel nicht vorhanden.



`c16_keyfldinfo(aConnection : resource, aFileNo : int, aKeyNo : int[, aKeyFldNo : int]) : mixed`  
 Informationen über ein Schlüsselfeld ermitteln

`aConnection` Verbindungsobjekt

`aFileNo` Dateinummer

`aKeyNo` Schlüsselnummer

`aKeyFldNo` Nummer des Schlüsselfeldes  
 (optional)

Resultat `mixed` Informationsarray oder Fehlerwert

Siehe [Befehle der PHP-Schnittstelle, `c16\_keyinfo\(\)`](#)

Der Befehl füllt einen Array mit Informationen über ein oder alle Schlüsselfelder.

Im Parameter (`aConnection`) wird das von [`c16\_connect\(\)`](#) bereitgestellte Verbindungsobjekt übergeben. In den Parametern (`aFileNo`) und (`aKeyNo`) müssen sich die Dateinummer und die Schlüsselnummer des Schlüssels befinden, von dem die Informationen abgefragt werden sollen. Wird keine Schlüsselfeld übergeben, wird ein indiziertes Array mit allen Schlüsselfeldern zurückgegeben. Ist in (`aKeyFldNo`) ein Schlüsselfeld angegeben, wird ein assoziatives Array mit folgenden Schlüsselwerten zurückgegeben:

`_FileNumber` Nummer der Datei

`_SbrNumber` Nummer des Teildatensatzes

`_FldNumber` Nummer des Schlüsselfeldes

`_FldName` Name des Schlüsselfeldes

`_FldType` Feldtyp

`_KeyFldAttributes` Attribute des Schlüsselfeldes

`_KeyFldMaxLen` Die definierte Maximallänge des Schlüsselfeldes

Der Rückgabewert des Feldtyps kann mit folgenden Konstanten verglichen werden:

#### Wert Konstante

1 `_TypeAlpha`

2 `_TypeDate`

3 `_TypeByte`

4 `_TypeWord`

5 `_TypeDecimal`

6 `_TypeMemo`

7 `_TypeInt`

8 `_TypeBigInt`

9 `_TypeFloat`

10 `_TypeLogic`

11 `_TypeTime`

Der Rückgabewert in `KeyFldAttributes` setzt sich aus Kombinationen der folgenden Werte zusammen:

## Kontakt

0x02 _KeyFldAttrUpperCase	Großschreibung
0x04 _KeyFldAttrUmlaut	Umlaute in alphabetischer Sortierung
0x08 _KeyFldAttrSpecialChars	ohne Sonderzeichen
0x10 _KeyFldAttrSoundex1	Soundex Stufe 1
0x20 _KeyFldAttrSoundex2	Soundex Stufe 2
0x40 _KeyFldAttrReverse	absteigende Sortierung

Bei einem Fehler wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_KEY (-604)	Schlüssel nicht vorhanden.
C16ERR_NO_KEY_FLD (-605)	Schlüsselfeld nicht vorhanden.

`c16_linkinfo(aConnection : resource, aFileNo :  
int[, aLinkNo : int]) : array`  
Informationen über eine Verknüpfung ermitteln  
aConnection Verbindungsobjekt

aFileNo Dateinummer  
aLinkNo Verknüpfungsnummer (optional)  
Resultat array Informationsarray oder  
Fehlerwert

Siehe Befehle der PHP-Schnittstelle,  
c16\_linkfldinfo()

Der Befehl gibt einen Array mit Informationen zu Verknüpfungen zurück.

Im Parameter (aConnection) wird das von `c16_connect()` bereitgestellte Verbindungsobjekt übergeben. In den Parametern (aFileNo) muss die Dateinummer übergeben werden. Wird (aLinkNo) nicht übergeben, wird ein indizierter Array mit den Verknüpfungsnummern als Index und den Verknüpfungsnamen als Wert zurückgegeben. Wird eine gültige Verknüpfungsnummer übergeben, wird ein assoziatives Array mit folgenden Informationen zurückgegeben:

Schlüsselwert	Beschreibung
_FileNumber	Nummer der Datei, in der die Verknüpfung definiert ist
_LinkNumber	Verknüpfungsnummer
_LinkName	Name der Verknüpfung
_LinkFldCount	Anzahl der Verknüpfungsfelder
_DestFileNumber	Nummer der verknüpften Datei
_DestKeyNumber	Schlüsselnummer der verknüpften Datei

Bei einem Fehler wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen `c16_error()` oder `c16_errortext()` ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_LINK (-606)	Verknüpfung nicht vorhanden.

c16\_linkfldinfo (aConnection : resource, aFileNo : int, aLinkNo : int[, aLinkFldNo : int]) : array

Information über ein Verknüpfungsfeld ermitteln

aConnection Verbindungsobjekt

aFileNo Dateinummer

aLinkNo Nummer der Verknüpfung

aLinkFldNo Nummer des Verknüpfungsfeldes  
(optional)

Resultat array Informationsarray oder  
Fehlerwert

Siehe Befehle der PHP-Schnittstelle,  
c16\_linkinfo()

Der Befehl füllt einen Array mit Informationen über ein oder alle Verknüpfungsfelder.

Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. In den Parametern (aFileNo) und (aLinkNo) müssen sich die Dateinummer und die Verknüpfungsnummer der Verknüpfung befinden, von dem die Informationen abgefragt werden sollen. Wird kein Verknüpfungsfeld übergeben, wird ein indiziertes Array mit allen Verknüpfungsfeldern zurückgegeben. Ist in (aLinkFldNo) ein Verknüpfungsfeld angegeben, wird ein assoziatives Array mit folgenden Schlüsselwerten zurückgegeben:

<b>Komponente</b>	<b>Beschreibung</b>
_FileNumber	Nummer der Datei, in der das Verknüpfungsfeld definiert ist
_SbrNumber	Nummer des Teildatensatzes, in dem das Verknüpfungsfeld definiert ist
_FldNumber	Nummer des Verknüpfungsfeldes
_FldName	Name des Verknüpfungsfeldes
_FldTyp	Feldtyp
_LinkFldAttributes	Attribute des Verknüpfungsfeldes
_LinkFldMaxLen	Die definierte Maximallänge des Verknüpfungsfeldes

Der Rückgabewert des Feldtyps kann mit folgenden Konstanten verglichen werden:

#### **Wert Konstante**

- 1    \_TypeAlpha
- 2    \_TypeDate
- 3    \_TypeByte
- 4    \_TypeWord
- 5    \_TypeDecimal
- 6    \_TypeMemo
- 7    \_TypeInt
- 8    \_TypeBigInt
- 9    \_TypeFloat
- 10   \_TypeLogic
- 11   \_TypeTime

## Kontakt

Der Rückgabewert in LinkFldAttributes setzt sich aus Kombinationen der folgenden Werte zusammen:

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
0x40 _LinkFldAttrPosition	nur Zugriffspositionierung

Bei einem Fehler wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.
C16ERR_NO_LINK (-606)	Verknüpfung nicht vorhanden.
C16ERR_NO_LINK_FLD (-607)	Verknüpfungsfeld nicht vorhanden.

c16\_recinfo(aConnection : resource, aFileNo  
: int, aInfoType : int[, aKeyNoOrSel : int]) :  
int

Informationen zu einem Datensatz ermitteln

aConnection Verbindungsobjekt

aFileNo Dateinummer

aInfoType Informationstyp

aKeyNoOrSel Schlüsselnummer oder  
Selektionsobjekt (optional)

Resultat int Information oder  
Fehlerwert

Siehe Befehle der PHP-Schnittstelle

Der Befehl ermittelt Informationen zu einem Datensatz. Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Im Parameter (aFileNo) wird die Dateinummer angegeben für welche die Informationen abgefragt werden sollen. Als Informationstyp stehen folgende symbolische Konstanten zur Verfügung:

_RecCount	Anzahl der Datensätze in der Datei
_RecID	Interne ID des aktuell geladenen Satzes
_RecLen	Die effektive Länge des im Hauptspeicher befindlichen Datensatzes. Das Resultat gibt die Größe des Datensatzes in Bytes zurück. In der effektiven Länge sind alphanumerische Felder nur mit ihrer aktuellen Länge berücksichtigt.
_RecLockedBy	Wird auf einen gesperrten Datensatz zugegriffen, kann hiermit die ID-Nummer des Benutzers ermittelt werden, der den Datensatz gesperrt hat.
_RecGetPos	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 (aKeyNoOrSel) angegeben.
_RecGetPosReverse	Analog zu _RecGetPos, die Position des aktuellen Satzes innerhalb der Datei wird jedoch nach umgekehrter Schlüsselsortierung ermittelt.

Die Angabe der Schlüsselnummer in (aKeyNoOrSel) muss nur erfolgen, wenn die Position des aktuellen Satzes ermittelt werden soll. Sollen Informationen aus einer Selektionsmenge ermittelt werden, muss in (aKeyNoOrSel) das vom Befehl c16\_selopen() bereitgestellte Selektionsobjekt übergeben werden. Dies ist nur bei den Informationen \_RecCount und \_RecGetPos relevant.

Die Information wird von der Anweisung zurückgegeben.

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_SEL_HDL_INVALID (-208)	Selektionsobjekt aKeyNoOrSel ungültig. Möglicherweise wurde die Selektion noch nicht durchgeführt.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.

## Kontakt

C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aKeyNoOrSel übergeben.
C16ERR_NO_FILE (-601)	Datei nicht vorhanden.

c16\_reclinkinfo(aConnection : resource, aSrcFileNo : int, aDstFileNo : int, aLinkNoOrSel : int, aInfoType : int) : int

Informationen zu verknüpften Datensätzen ermitteln

aConnection Verbindungsobjekt

aSrcFileNo Nummer der Quell-Datei

aDstFileNo Nummer der Ziel-Datei

aLinkNoOrSel Nummer der Verknüpfung oder Selektionsobjekt

aInfoType Informationstyp

Resultat int Information oder Fehlerwert

Siehe Befehle der PHP-Schnittstelle

Der Befehl ermittelt Informationen zu einem verknüpften Datensatz. Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Im Parameter (aSrcFileNo) wird die Nummer der Ausgangsdatei, in (aDstFileNo) die Nummer der Zieldatei und in (aLinkNoOrSel) die Nummer der Verknüpfung angegeben, für welche die Informationen abgefragt werden sollen. Als Informationstyp stehen folgende symbolische Konstanten zur Verfügung:

_RecCount	Anzahl der verknüpften Datensätze in der Zieldatei. Eine eventuelle Zugriffspositionierung wird dabei ignoriert.
_RecGetPos	Die Position des aktuellen Satzes der Zieldatei. Der erste verknüpfte Satz hat dabei die Nummer 1. Das Resultat ist -1, wenn der Satz nicht zur verknüpften Menge gehört.
_RecCountPos	Anzahl aller verknüpften Datensätze ab der definierten Zugriffsposition. Der per Zugriffspositionierung verknüpfte Satz wird mitgezählt.
_RecCountNext	Die Anzahl verknüpfter Sätze nach dem aktuellen Satz in der Zieldatei. Das Resultat ist -1, wenn der Satz nicht zur verknüpften Menge gehört.

Der Rückgabewert vom Typ int beinhaltet die entsprechende Information.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_SEL_HDL_INVALID (-208)	Selektionsobjekt ungültig. Möglicherweise wurde die Selektion noch nicht durchgeführt.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_TYPE (-252)	Typ eines mehrtypigen Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aLinkNoOrSel übergeben.
C16ERR_NO_FILE (-601)	Die übergebene Dateinummer existiert nicht.



## Kontakt

C16ERR\_NO\_LINK (-606)

Die übergebene Verknüpfung existiert nicht.

C16ERR\_LINK\_INVALID (-609)

Verknüpfung aLinkNoOrSel ungültig.

c16\_selinfo(aSelection : resource[,  
aInfoType : string]) : mixed  
Selektionsinformationen ermitteln  
aSelection Selektionsobjekt

aInfoType zu ermittelnde  
Information (optional)  
Rückgabewert  
Resultat mixed mit der  
Information oder  
Fehlerwert

Siehe Befehle der  
PHP-Schnittstelle

Diese Funktion ermittelt alle Informationen zu einer Selektion. In (aSelection) wird das von c16\_selopen() bereitgestellte Selektionsobjekt übergeben.

Wird kein (aInfoType) übergeben, gibt der Befehl ein assoziatives Array mit folgenden Informationen zurück:

#### **Schlüssel Wert**

_SelFile	Dateinummer der Selektion
_SelName	Name der Selektion
_SelSort	Schlüsselnummer der Selektionssortierung
_SelCount	Anzahl der Werte
_SelCountD	Anzahl der verschiedenen Werte
_SelMin	kleinster Wert
_SelMax	größter Wert
_SelSum	Summe der Werte
_SelSumD	Summe der verschiedenen Werte
_SelAvg	Durchschnitt der Werte
_SelAvgD	Durchschnitt der verschiedenen Werte

Die Informationen können einzeln abgefragt werden, wenn in (aInfoType) die entsprechende Konstante angegeben wird. Der Typ des Rückgabewerts hängt von der angeforderten Information ab. Bei Selektionen ohne Wertemenge stehen nur die ersten drei Informationen zur Verfügung.

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_VALUE (-253)	Wert eines Arguments ungültig.
C16ERR_ARGS_SELECTION (-255)	Kein Selektionsobjekt in aSelection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.

## PHP-Schnittstelle - Satzpuffer-Funktionen

Übersicht über die Satzpuffer-Funktionen der PHP-Schnittstelle

- c16\_fldget
- c16\_fldset

c16\_fldget(aConnection :  
resource, aFieldArray : array) :  
int  
Feldpuffer in Array übertragen  
aConnection Verbindungsobjekt  
aFieldArray Array für die  
Feldpufferinhalte  
Resultat int Fehlerwert  
Siehe Befehle der  
PHP-Schnittstelle,  
c16\_fldset()

Mit dieser Anweisung wird der Inhalt des Datensatzpuffers in das Array übertragen. Anschließend kann der Inhalt des Arrays ausgewertet werden.

Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Das in (aFieldArray) übergebene Array kann ein assoziatives oder indiziertes Array sein. Die Schlüsselwerte des Arrays definieren dabei entweder den Namen des Feldes oder die Nummer des Feldes.

### Beispiel:

Das folgende Beispiel gibt alle Datensätze einer Datei aus. Ausgegeben werden nur die Felder "CST.iNumber" und "CST.aDescription", die in dem Array angegeben werden.



Damit die Konstanten verwendet werden können, muss die Datei "php\_c16\_def.php" mit der Anweisung `include('php_c16_def.php');` eingebunden werden.

```
echo '<table>'; // define array with fields$fields = array('ART.iNumber' => 0, 'ART.aBezeichnung'
```

Über den Rückgabewert der Funktion kann ermittelt werden, ob die Funktion korrekt ausgeführt wurde. Der Rückgabewert kann mit folgenden Konstanten verglichen werden:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_NO_FLD (-603)	Feld nicht vorhanden.

c16\_fldset(aConnection : resource,  
aFieldArray : array) : int  
Array-Inhalt in Feldpuffer übertragen  
aConnection Verbindungsobjekt  
aFieldArray Zu übertragendes  
Array  
Resultat int Fehlerwert  
Siehe Befehle der  
PHP-Schnittstelle,  
c16\_fldget()

Mit dieser Anweisung wird der Inhalt des Arrays in die Feldpuffer der Datenbank übertragen. Anschließend kann eine der Datensatz-Funktionen ausgeführt werden.

Im Parameter (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Das in (aFieldArray) übergebene Array kann ein assoziatives oder indiziertes Array sein. Die Schlüsselwerte des Arrays definieren dabei entweder den Namen des Feldes oder die Nummer des Feldes.

Sollen Werte für Felder definiert werden, deren Datentypen nicht in PHP existieren, müssen diese Werte in einer Zeichenkette im CONZEPT 16 eigenen Format angegeben werden. Ein Datums-Wert wird in der Form 'dd.mm.yyyy', ein Zeitwert in der Form 'hh.mm.ss' angegeben. Bei der Übergabe von Werten in ein Feld vom Typ bigint werden alle nicht-numerischen Zeichen innerhalb der Zeichenkette ignoriert. Bei decimal muss der Punkt als Dezimal-Trennzeichen verwendet werden.

### Beispiel:

Das folgende Beispiel gibt alle Datensätze einer Datei aus. Ausgegeben werden nur die Felder fiArtNumber und faArtName, die in dem Array angegeben werden.



Damit die Konstanten verwendet werden können, muss die Datei php\_c16\_def.php mit der Anweisung include('php\_c16\_def.php'); eingebunden werden.

```
echo '<table>';// define array with fields$fields = array('fiArtNumber' => 0, 'faArtName' => '');
```

Über den Rückgabewert der Funktion kann ermittelt werden, ob die Funktion korrekt ausgeführt wurde. Der Rückgabewert kann mit folgenden Konstanten verglichen werden:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_NO_FLD (-603)	Feld nicht vorhanden.



## PHP-Schnittstelle - Sonstige Funktionen

Übersicht über die sonstigen Funktionen der PHP-Schnittstelle

- [c16\\_call](#)
- [c16\\_error](#)
- [c16\\_errortext](#)
- [c16\\_localeselect](#)
- [c16\\_sysparamset](#)


c16\_call(aConnection :  
resource, aProcName : string[[,  
aParameter : mixed]]) : mixed  
CONZEPT 16-Funktion aufrufen  
aConnection Verbindungsobjekt  
aProcName Prozedurname in der  
Datenbank  
aParameter Parameter (optional)  
Resultat mixed Rückgabewert  
der Prozedur  
Siehe Befehle der  
PHP-Schnittstelle

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 (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Der Prozedurname wird im Parameter (aProcName) übergeben. Der Name kann sich dabei auch aus <Prozedurname:Funktionsname> zusammensetzen.

 Von der PHP-Schnittstelle können nur Prozeduren ausgeführt werden, die mit der Option @A+ übersetzt wurden. Alle Befehle, die in diesen Prozeduren verwendet werden können, sind in der Online Hilfe mit dem Symbol  versehen. Ab dem dritten Parameter können die Übergabewerte an die Prozedur übergeben werden. Folgende PHP-Typen können übergeben werden:

### PHP-Typ CONZEPT 16-Typ

array int  
string alpha  
integer int  
double float  
boolean logic

 Es können keine VAR-Parameter (Call by reference) angegeben werden.

Wird eine Zeichenkette oder ein Array übergeben, darf die Zeichenkette und die Elemente des Arrays nicht größer als 4 KB sein. Wird ein Array übergeben, muss im Funktionskopf der CONZEPT 16-Funktion eine ganzzahlige Variable definiert werden. In dieser Variablen wird der Deskriptor einer Cte-Liste übergeben. Der Inhalt des Arrays befindet sich in den Cte-Items der übergebenen Liste. Bei einem indizierten Array sind die Schlüssel in der Eigenschaft ID, bei einem assoziativen Array in der Eigenschaft Name abgelegt. Der Wert befindet sich in der Eigenschaft Custom.

Als Rückgabewert der CONZEPT 16-Funktion können die gleichen Typen verwendet werden, die auch übergeben werden können. Soll ein Array zurückgegeben werden, muss der Rückgabewert der Funktion mit handle angegeben werden.

### Beispiele:

### Definition in CONZEPT 16

## Kontakt

```
sub Alphabet( aArg1 : alpha; aArg2 : float; aArg3 : int;) : handle;{ ...}
```

### Aufruf in PHP

```
$arg1 = 'String';$arg2 = 3.14159;$arg3 = array( 1 =>'abc', 2 => 'def');$arg4 = array( 'Vowels' =>
```

Tritt während der Verarbeitung der Prozedur ein Laufzeitfehler auf, bricht die Prozedur ab. Der Fehler kann mit Hilfe der Funktion c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_CONNECTION (-254)	Kein Verbindungsobjekt in aConnection übergeben.
C16ERR_RESULT_ARRAY (-256)	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.



c16\_localeselect(aCategory : int[,  
aLocale : string])  
Formatierung für Typenwandlung setzen  
aCategory Kategorie

aLocale zu aktivierende  
Ländereinstellung

Resultat int Fehlerwert

Siehe [Befehle der  
PHP-Schnittstelle](#)

Mit dieser Funktion können die landesspezifischen Parameter bei der automatischen Typenkonvertierung der CONZEPT 16-Datentypen decimal, bigint, date und time eingestellt werden. Standardmäßig wird in das interne CONZEPT 16-Format gewandelt.

In (aCategory) wird bestimmt, welche Kategorie von der mit der in (aLocale) definierten Ländereinstellung umgewandelt werden. Folgende Konstanten können angegeben werden:

#### **Konstante Kategorie**

\_LcInternal alle Werte auf CONZEPT 16-Format setzen

\_LcAll alle Werte

\_LcMonetary Währungswerte

\_LcNumeric Numerische Werte

\_LcTime Datum- und Zeitwerte

Bei der Angabe der Kategorie \_LcInternal darf keine Ländereinstellung angegeben werden. Die Wandlung erfolgt dann im CONZEPT 16-eigenen Format. Bei allen anderen Kategorien muss eine Ländereinstellung in (aLocale) angegeben werden.

#### **Beispiele:**

```
// Wandlung im internen Formatc16_localeselect(_LcInternal);// Wandlung und Darstellung im schwei
```

Bei einem Fehler wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_VALUE (-253)	Wert eines Arguments ungültig.

c16\_sysparamset(aParameter : int, aValue :  
string) : int  
Setzen eines Wertes für die Typenkonvertierung

aParameter    Zu verändernder  
Wert  
aValue        Neuer Wert nach  
einer  
Typkonvertierung  
Resultat      int Fehlerwert  
Siehe         Befehle der  
               PHP-Schnittstelle

Mit dieser Funktion können bei der automatischen Typenkonvertierung bestimmte Werte gesetzt werden.

Für folgende Werte können Konvertierungswerte angegeben werden:

### **Wert in CONZEPT 16 (aValue)**

true                    \_SysParFmtBoolTrue  
false                    \_SysParFmtBoolFalse

### **Beispiele:**

```
// Setzt für true/false die Werte TRUE/FALSEc16_sysparamset(_SysParFmtBoolTrue , 'TRUE');c16_sysp
```

Tritt bei der Verarbeitung ein Fehler auf, wird der Fehlerwert von der Anweisung zurückgegeben. Der Fehler kann ebenfalls mit den Anweisungen c16\_error() oder c16\_errortext() ermittelt werden. Folgende Fehler können auftreten:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_VALUE (-253)	Wert eines Arguments ungültig.

c16\_error(aConnection :  
resource) : int  
Ermitteln des Fehlerwertes  
aConnection Verbindungsobjekt

Resultat     int Fehlerwert

Siehe         Verwandte  
              Befehle,  
              c16\_errortext(),  
              Fehlerwerte

Mit dieser Anweisung wird der Fehlerwert einer CONZEPT 16-Anweisung in einem PHP-Skript nach der Durchführung der Anweisung ermittelt. In (aConnection) wird das von c16\_connect() bereitgestellte Verbindungsobjekt übergeben. Der Rückgabewert kann mit den Fehlerkonstanten (siehe Fehlerwerte) verglichen werden.

Da der Fehlerwert erst nach der Durchführung der Anweisung ermittelt werden kann, kann c16\_error() nicht nach der Anweisung c16\_close() verwendet werden, da in der Regel das Verbindungsobjekt nicht mehr vorhanden ist. Die Anweisung liefert einen Fehlerwert zurück, der Aufruf der Anweisung c16\_error() ist daher nicht notwendig.

Von der Anweisung können folgende Resultatwerte zurückgegeben werden:

C16_OK (0)	Kein Fehler aufgetreten.
C16ERR_ARGS_FORMAT (-251)	Anzahl der Argumente oder Typ eines Arguments ungültig.
C16ERR_ARGS_CONNECTION (-254)	Verbindungsobjekt in aConnection ungültig.

`c16_errortext([aErrorSource  
: mixed]) : mixed`

Fehlertext ermitteln

`aErrorSource` Fehlerquelle  
(optional)

Resultat mixed Fehlertext

Siehe Befehle der  
PHP-Schnittstelle,  
`c16_error()`,  
Fehlerwerte

Mit dieser Funktion können die Klartexte zu Fehlern ermittelt werden. Folgende Anwendungen stehen zur Verfügung:

- **Alle Fehlermeldungen ermitteln**

Wird der Befehl ohne Parameter aufgerufen, wird ein Array mit allen Fehlermeldungen zurückgegeben. Auf den Inhalt des indizierten Array kann mit Hilfe der Fehlerkonstanten zugegriffen werden.

```
$errortext = c16_errortext();echo $errortext[C16ERR_SERVER_USER_LIMIT];
```

- **Fehlertext zu einer Fehlernummer ermitteln**

Wird der Befehl mit einer Fehlernummer aufgerufen, wird der Klartext zu diesem Fehler in einer Zeichenkette zurückgegeben.

```
echo c16_errortext(C16ERR_SERVER_USER_LIMIT);
```

- **Fehlertext des zuletzt aufgetretenen Fehlers ermitteln**

Wird der Befehl mit einem Verbindungsobjekt (siehe `c16_connect()`) aufgerufen, wird der Klartext zu dem zuletzt aufgetretenen Fehler als Zeichenkette zurückgegeben.

```
$connection = c16_connect('TCP:10.1.0.2+10.1.0.1', '', 'Examples', 'USER', '');if (c16_err
```

Von der Anweisung können folgende Resultatwerte zurückgegeben werden:

<code>C16_OK (0)</code>	Kein Fehler aufgetreten.
<code>C16ERR_ARGS_FORMAT (-251)</code>	Anzahl der Argumente oder Typ eines Arguments ungültig.
<code>C16ERR_ARGS_TYPE (-252)</code>	Typ eines mehrtypigen Arguments ungültig.
<code>C16ERR_ARGS_CONNECTION (-254)</code>	Kein Verbindungsobjekt übergeben.
<code>C16ERR_RESULT_ARRAY (-256)</code>	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.

Fehlerwerte (PHP-Schnittstelle)

Rückgabewerte der Funktionen der PHP-Schnittstelle

Alle Funktionen der CONZEPT 16-PHP-Schnittstelle setzen einen Fehlerwert. Der Fehlerwert kann mit der Funktion c16\_error() ermittelt und mit folgenden Konstanten verglichen werden:

<b>Code</b>	<b>Symbolische Konstante</b>	<b>Fehlertext</b>	<b>Kurzbeschreibung</b>
0	C16_OK		Kein Fehler aufgetreten.
-1	C16ERR_GENERIC	generic error	Allgemeiner Fehler aufgetreten.
-208	C16ERR_SEL_HDL_INVALID	selection invalid	Selektionsobjekt ungültig. Möglicherweise wurde die Selektion noch nicht durchgeführt.
-251	C16ERR_ARGS_FORMAT	wrong argument format	Anzahl der Argumente oder Typ eines Arguments ungültig.
-252	C16ERR_ARGS_TYPE	wrong argument type	Typ eines mehrtypigen Arguments ungültig.
-253	C16ERR_ARGS_VALUE	wrong argument value	Wert eines Arguments ungültig.
-254	C16ERR_ARGS_CONNECTION	wrong argument: connection	Kein Verbindungsobjekt übergeben.
-255	C16ERR_ARGS_SELECTION	wrong argument: selection	Kein Selektionsobjekt übergeben.
-256	C16ERR_RESULT_ARRAY	result array initialisation failed	Initialisierung eines zurückzugebenden Arrays fehlgeschlagen.
-302	C16ERR_COMM_FAILED	communication failed	Kommunikation fehlgeschlagen.
-303	C16ERR_SERVER_OP	server operation failed	Serverseitige Verarbeitung fehlgeschlagen.
-305	C16ERR_SWAPINIT_FAILED	failed to initialize swap file	Initialisierung der Auslagerungsdatei fehlgeschlagen. Möglicherweise ist in <u>c16.path.temp</u> ein ungültiges Verzeichnis

-401	C16ERR_NO_SERVER_CONNECTION	no server connection	angegeben. Serververbindung konnte nicht hergestellt werden.
-402	C16ERR_AREA_NOT_FOUND	database not found	Datenbank nicht gefunden.
-403	C16ERR_AREA_OPEN	database open failed	Öffnen der Datenbank fehlgeschlagen.
-404	C16ERR_AREA_LOCKED	database is locked	Datenbank gesperrt.
-405	C16ERR_AREA_IN_USE	database in exclusive use	Datenbank in exklusiver Benutzung.
-406	C16ERR_AREA_TYPE	invalid database type	Datenbank nicht kompatibel.
-407	C16ERR_AREA_PASSWORD	invalid server password	Serverpasswort nicht korrekt.
-408	C16ERR_SERVER_USER_LIMIT	user limit is reached (server)	Benutzerlimit des Servers erreicht.
-409	C16ERR_SERVER_START	cannot start server	Serverprozess konnte nicht gestartet werden.
-413	C16ERR_USER_INVALID	user authorization failed	Benutzername oder -passwort nicht korrekt.
-416	C16ERR_AREA_STANDBY	server standby mode	Datenbank im Standby-Modus.
-417	C16ERR_AREA_ROLLBACK	database in rollback mode	Datenbank im Rollback-Modus.
-418	C16ERR_AREA_LOCKED_ADMIN	Database is locked by administrator	Datenbank durch Administrator gesperrt. <u>siehe Fehler beim Öffnen der Datenbank</u>
-419	C16ERR_AREA_LOCKED_OPERATION	Database is locked by service operation	Datenbank durch Serviceoperation gesperrt. <u>siehe Fehler beim Öffnen der Datenbank</u>
-420	C16ERR_AREA_LOCKED_DOWN	Database is locked by shutdown	Datenbank wird geschlossen. <u>siehe Fehler beim Öffnen der</u>

-421	C16ERR_AREA_LOCKED_STANDBY	Database is locked (standby mode)	<u>Datenbank</u> Datenbank durch Standby-Modus gesperrt. siehe <u>Fehler beim Öffnen der Datenbank</u>
-422	C16ERR_AREA_LOCKED_ROLLBACK	Database is locked (rollback denied)	Datenbank durch Rollback-Sperre gesperrt. siehe <u>Fehler beim Öffnen der Datenbank</u>
-423	C16ERR_AREA_LOCKED_OPEN	Database is locked (login denied)	Datenbank durch Login-Sperre gesperrt. siehe <u>Fehler beim Öffnen der Datenbank</u>
-424	C16ERR_AREA_LOCKED_NO_STANDBY_OPEN	Database is locked (standby system not available)	Datenbank durch Standby-Sperre gesperrt. siehe <u>Fehler beim Öffnen der Datenbank</u>
-600	C16ERR_NO_DATASTRUCT	no datastruct	Datenstruktur konnte nicht geladen werden.
-601	C16ERR_NO_FILE	no file	Datei nicht vorhanden.
-602	C16ERR_NO_SBR	no subrecord	Teildatensatz nicht vorhanden.
-603	C16ERR_NO_FLD	no field	Feld nicht vorhanden.
-604	C16ERR_NO_KEY	no key	Schlüssel nicht vorhanden.
-605	C16ERR_NO_KEY_FLD	no key field	Schlüsselfeld nicht vorhanden.
-606	C16ERR_NO_LINK	no link	Verknüpfung nicht vorhanden.
-607	C16ERR_NO_LINK_FLD	no link field	Verknüpfungsfeld nicht vorhanden.
-609	C16ERR_LINK_INVALID	link invalid	Verknüpfung ungültig.
-2001	C16ERR_LICENSE_SRVNOTSUPPORTED	server release not supported	Version des Datenbankservers nicht unterstützt.

## Kontakt

-2002	C16ERR_LICENSE_CLNUPGRADE	client upgrade missing	Version des Clients nicht lizenziert.
-2003	C16ERR_LICENSE_CLNNOTSUPPORTED	insufficient client license	Typ des Clients nicht lizenziert.
-2004	C16ERR_LICENSE_COMM	license communication error	Kommunikation mit Lizenzserver fehlgeschlagen.
-2005	C16ERR_LICENSE_DATA	license transmission error	Datenaustausch mit Lizenzserver fehlgeschlagen.
-2006	C16ERR_LICENSE_INVALID	license error occurred	Lizenzdaten ungültig.
-2007	C16ERR_LICENSE_MISMATCH_EVAL	license mismatch evaluation	Client nicht von Evaluierungslizenz unterstützt.

Wird beim Ausführen einer Prozedur ein Laufzeitfehler erzeugt, bricht die Prozedur ab und ein Fehlerwert wird erzeugt, der dem CONZEPT 16-Laufzeitfehler verkleinert um 1000 entspricht. Der Fehler -1170 entspricht also dem Laufzeitfehler -170 ([\\_ErrNoProcInfo](#)).

Der englische Klartext zu einer Fehlernummer kann mit der Anweisung [c16\\_errortext\(\)](#) ermittelt werden.



## Plugin-Schnittstelle

Steuerung des Designers durch externe Anwendungen

Die Designer Plugin-Schnittstelle erweitert den Designer um die Möglichkeit, externe Anwendungen (Plugin-Clients) an den Designer anzubinden. Der Plugin-Client verbindet sich hierzu via Socket mit dem Designer. Nach erfolgreicher Authentifikation kann der Plugin-Client sowohl Befehle an den Designer senden, als auch Ereignisse des Designers empfangen. Somit ist es möglich den Designer um eigene Funktionen zu erweitern und ihn zu individualisieren.

In der CodeLibrary ist ein Beispiel zur Anwendung der Plugin-Schnittstelle enthalten. In diesem Beispiel wird in einem Dialog eine Liste aller Dialoge angezeigt. Wird ein Dialog im Designer aktiviert, wird dieser auch in der Liste selektiert. Durch einen Doppelklick auf einen Eintrag wird der Dialog im Designer geöffnet.

### 1. Funktionsweise

#### 1. Nachrichtenaustausch

##### 1. Genereller Aufbau des Nachrichtenformates

1. Antwortkennung
2. Befehlskennung
3. Befehlsname
4. Argumentliste
5. Rückgabe-Argumentliste

##### 2. Beispiele

1. Befehl
2. Ereignis
3. Antwort

##### 2. Verbindungsherstellung

##### 3. Authentifikation

### 2. Erstellung einer Plugin-Anwendung in CONZEPT 16

#### 1. Core-API

##### 1. Instanziierung

1. Erstellung einer Instanz
2. Schließen einer Instanz
3. Schließen aller Instanzen
4. Iterieren der Instanzen
5. Ermitteln des JobControl-Objektes
6. Senden einer Nachricht
7. Empfangen einer Nachricht
8. Eindeutige Antwortkennung generieren
9. Informationen zur API ermitteln
10. Ermitteln des zuletzt aufgetretenen Fehlers

#### 2. Converter-API

1. Authentifizierungsaufforderung erwarten
2. Authentifizierungsaufforderung beantworten
3. Erstellen eines neuen PluginCommand-Handles
4. Löschen eines PluginCommand-Handles
5. Leeren eines PluginCommand-Handles
6. Name des Plugin-Kommandos ermitteln
7. Name eines Plugin-Kommandos überprüfen
8. Ermitteln des Kommando-Typs
9. Überprüfung auf einen bestimmten Kommando-Typ

10. Ermitteln der Antwortkennung (fortlaufende Nummer)
  11. Ermitteln des zuletzt aufgetretenen Fehlers
  12. Setzen von Argumenten
    1. String-Argument
    2. 32-bit vorzeichenbehaftetes ganzzahliges Argument
    3. Logisches Argument
    4. String-Argument als Memory-Objekt
    5. Rückgabe-Argument
    6. Argument für Resultat des Designer-Befehls
  13. Ermitteln von Argumenten
    1. Argument nach Name ermitteln
    2. Argument nach Nummer ermitteln
    3. Ermitteln der Anzahl der Argumente
    4. Ermitteln der Länge eines Argumentes
    5. Argumentwert vom Typ \_TypeAlpha ermitteln
    6. Argumentwert eines Base64 kodierten Argumentes vom Typ \_TypeAlpha ermitteln
    7. Argumentwert vom Typ \_TypeInt ermitteln
    8. Argumentwert vom Typ \_TypeLogic ermitteln
    9. String-Argument als Memory-Objekt ermitteln
  14. Kodieren / dekodieren von Kommandos
    1. Kodieren eines Plugin-Kommandos in das Nachrichtenformat
    2. Dekodieren eines Plugin-Kommandos aus dem Nachrichtenformat
    3. Empfangen und dekodieren einer Nachricht
    4. Kodieren und Senden einer Nachricht
3. Ereignisse des Designers
1. Designer.Forms.OpenDone
  2. Designer.Forms.CloseDone
  3. Designer.Forms.ActivateDone
  4. Designer.Forms.SaveDone
  5. Designer.Forms.WriteLockChangeDone
  6. Designer.Forms.Selection.Menu.MenuSelect
  7. Designer.Dictionary.Enter
  8. Designer.Dictionary.Exit
  9. Designer.Editor.OpenDone
  10. Designer.Editor.CloseDone
  11. Designer.Editor.ActivateDone
  12. Designer.Editor.CompileDone
  13. Designer.Editor.SaveDone
  14. Designer.Editor.Document.Menu.MenuSelect
  15. Designer.MenuEditor.OpenDone
  16. Designer.MenuEditor.SaveDone
  17. Designer.Menu.Plugins.MenuSelect
  18. Designer.Exit
4. Designer-Befehle
1. Designer.Auth
  2. Designer.GetInfo
  3. Designer.Forms.\*
    1. Designer.Forms.Open

2. **Designer.Forms.Export**
3. **Designer.Forms.Import**
4. **Designer.Forms.GetCount**
5. **Designer.Forms.GetObject**
6. **Designer.Forms.Selection.Get**
7. **Designer.Forms.Selection.New**
8. **Designer.Forms.Selection.GetCount**
9. **Designer.Forms.Selection.GetObject**
10. **Designer.Forms.Selection.GetPreview**
11. **Designer.Forms.Selection.Menu.AddItem**
12. **Designer.Forms.Selection.Menu.<Name>.Change**
13. **Designer.Forms.Selection.Menu.<Name>.Remove**
4. **Designer.MenuEditor.Open**
5. **Designer.Editor.\***
  1. **Designer.Editor.Open**
  2. **Designer.Editor.Compile**
  3. **Designer.Editor.Document.Menu.AddItem**
  4. **Designer.Editor.Document.Menu.<Name>.Change**
  5. **Designer.Editor.Document.Menu.<Name>.Remove**
6. **Designer.Menu.\***
  1. **Designer.Menu.File.New.Frame.Exec**
  2. **Designer.Menu.File.New.AppFrame.Exec**
  3. **Designer.Menu.File.New.MdiFrame.Exec**
  4. **Designer.Menu.File.New.TrayFrame.Exec**
  5. **Designer.Menu.File.New.PrintDoc.Exec**
  6. **Designer.Menu.File.New.PrintDocRecord.Exec**
  7. **Designer.Menu.File.New.PrintForm.Exec**
  8. **Designer.Menu.File.New.PrintFormList.Exec**
  9. **Designer.Menu.File.New.Menu.Exec**
  10. **Designer.Menu.File.Open.Exec**
  11. **Designer.Menu.File.Reopen.Exec**
  12. **Designer.Menu.File.Save.Exec**
  13. **Designer.Menu.File.SaveAs.Exec**
  14. **Designer.Menu.File.ReadWrite.Exec**
  15. **Designer.Menu.File.Close.Exec**
  16. **Designer.Menu.File.CloseAll.Exec**
  17. **Designer.Menu.File.Import.Exec**
  18. **Designer.Menu.File.Export.Exec**
  19. **Designer.Menu.File.Menu.Exec**
  20. **Designer.Menu.File.Test.Exec**
  21. **Designer.Menu.File.MainProc.Exec**
  22. **Designer.Menu.File.Config.Exec**
  23. **Designer.Menu.File.Exit.Exec**
  24. **Designer.Menu.Extras.Dictionary.Exec**
  25. **Designer.Menu.Extras.RecordManagement.Exec**
  26. **Designer.Menu.Extras.BlobManagement.Exec**
  27. **Designer.Menu.Extras.UserManagement.Exec**
  28. **Designer.Menu.Extras.Resource.Exec**
  29. **Designer.Menu.Extras.ViewAppLog.Exec**
  30. **Designer.Menu.Plugins.AddItem**
  31. **Designer.Menu.Plugins.<Name>.Change**

- 32. **Designer.Menu.Plugins.<Name>.Remove**
- 7. **Designer.Storage.\***
  - 1. **Designer.Storage.Create**
  - 2. **Designer.Storage.Load**
  - 3. **Designer.Storage.NewObject**
  - 4. **Designer.Storage.<StorageName>.Save**
  - 5. **Designer.Storage.<StorageName>.Unload**
  - 6. **Designer.Storage.<ObjectName>.DeleteObject**
  - 7. **Designer.Storage.<ObjectName>.RenameObject**
  - 8. **Designer.Storage.<ObjectName>.WriteKey**
  - 9. **Designer.Storage.<ObjectName>.ReadKey**
  - 10. **Designer.Storage.<ObjectName>.ReadNames**
  - 11. **Designer.Storage.LastLockingInfo**
  - 12. **Designer.Storage.<ObjectName>.ExportFile**
  - 13. **Designer.Storage.<ObjectName>.ExportStream**
  - 14. **Designer.Storage.ImportFile**
  - 15. **Designer.Storage.ImportStream**
- 8. **Designer.Plugins.Shutdown**
- 9. **Designer.LastXmlError**

## Plugin-Schnittstelle - Funktionsweise

### Funktionsweise der Plugin-Schnittstelle des Designers

Siehe Plugin-Schnittstelle,  
Befehle, Ereignisse,  
Plugin-Schnittstelle  
in CONZEPT 16

Die Plugin-Funktionalität des Designers wird aktiviert, wenn in der Konfigurationsdatei der Eintrag PluginPort mit einer gültigen Portnummer ( $\leq 65535$ ) eingetragen ist. Die Portnummer kann beim Start über die Kommandozeilenoption /c16PluginPort oder auch über die Umgebungsvariable C16PluginPort definiert werden.

Zusätzlich zu dem Port muss bei dem jeweiligen Benutzer des Designers noch ein Plugin-Kennwort festgelegt werden.

Der Designer baut nach dem Starten und vor dem Laden des Benutzerprofils eine passive Socket-Verbindung über die angegebene Portnummer auf. Danach können sich ein oder mehrere Plugin-Clients mit dem Designer verbinden.

Da die Kommunikation von Plugin-Client und Designer über ein textbasiertes Protokoll abläuft, können Plugin-Clients in jeder Sprache (C++, C#, etc.) entwickelt werden, die Sockets unterstützen. Für die Entwicklung von Plugin-Clients in CONZEPT 16 wurde eine Core-API entwickelt, welche die Entwicklung für Plugin-Anwendungen vereinfacht.



Fehler, die bei der Kommunikation auftreten werden im Anwendungsprotokoll des Designers festgehalten.

## Überblick

### 1. Nachrichtenaustausch

#### 1. Genereller Aufbau des Nachrichtenformates

1. Antwortkennung
2. Befehlskennung
3. Befehlsname
4. Argumentliste
5. Rückgabe-Argumentliste

#### 2. Beispiele

1. Befehl
2. Ereignis
3. Antwort

### 2. Verbindungsherstellung

### 3. Authentifikation

---

### 1. Nachrichtenaustausch

Der Nachrichtenaustausch von Plugin-Client zu Designer und umgekehrt läuft über ein textbasiertes Nachrichtenformat. Alle Zeichenketten sind hierbei UTF-8 kodiert. Zwischen Groß- und Kleinschreibung wird nicht unterschieden (außer bei Dateninhalten).

## Genereller Aufbau des Nachrichtenformates

Der Aufbau einer Nachricht hat den folgenden syntaktischen Aufbau:

```
[<serial> | ] <type> [<command-name>] (<arg-list>) [\<return-arg-list>\] ]
```

### 1. Antwortkennung (<serial>)

Bei <serial> handelt es sich um eine fortlaufende 64-bit ganzzahlige Nummer, die mit gesendet wird, wenn eine Antwort auf ein Kommando oder ein Ereignis erwartet wird. Die Nummer fehlt, wenn keine Antwort erwartet wird.

### 2. Befehlskennung (<type>)

Es gibt drei Befehlskennungen:

CMD Bei der Nachricht handelt es sich um einen Befehl.

EVT Bei der Nachricht handelt es sich um ein Ereignis.

RET Bei der Nachricht handelt es sich um die Antwort auf CMD oder EVT.

### 3. Befehlsname (<command-name>)

Sofern ein Kommando (CMD) oder ein Ereignis (EVT) gesendet wird, muss ein Befehlsname angegeben sein. Dieser gibt darüber Auskunft, welches Ereignis ausgelöst, bzw. welcher Befehl durchgeführt werden soll. Der Befehlsname wird in URL-Schreibweise angegeben (siehe [Befehlsbeschreibung](#)). Befehlsnamen können neben dem Punkt (.) nur aus Buchstaben, Ziffern und dem Unterstrich (\_) bestehen.

### 4. Argumentliste (<arg-list>)

Die Argumentliste enthält die zum Befehl bzw. Ereignis gehörenden Argumente. Die Argumentliste kann auch leer sein. In diesem Fall muss eine leere Argumentliste () angegeben werden.

```
<arg-list> = [<arg-name>=<arg-value>[[,<arg-name>=<arg-value>]]]
```

<arg-name> bezeichnet den Argumentname. Das erste Zeichen des Arguments muss ein Buchstabe sein (A...Z). Danach können sowohl Buchstaben also auch Ziffern, sowie der Unterstrich (\_) folgen.

<arg-value> enthält den Wert des durch <arg-name> angegebenen Arguments und kann folgende Werte annehmen:

```
<arg-value> = { <string-literal> | <number> | true | false | ? }
```

#### 1. String-Literal (<string-literal>)

Hierdurch wird eine UTF8-kodierte Zeichenkette angegeben. Diese muss in doppelten Hochkommata eingeschlossen sein.

1. `<string-literal> = "[[<char>]]"`

`<char>` ist jedes gültige UTF8-codierte Zeichen. Darüber hinaus gibt es reservierte Zeichen, die durch das Escape-Zeichen `\` maskiert werden müssen:

`\"` Doppelte Hochkommata  
`\n` Zeilenvorschub (Line feed)  
`\t` Tabulatorzeichen  
`\b` Backspace  
`\f` Seitenvorschub  
`\r` Zeilenrücklauf (Carriage return)  
`\\` Backslash  
`\/` Slash

## 2. Nummer (`<number>`)

Hierdurch wird ein positiver oder negativer ganzzahliger numerischer Wert (32-bit) angegeben.

## 3. Logische Werte (`true`, `false`)

Hierdurch wird ein Argument mit einem logischen Wert (`true` oder `false`) angegeben.

## 4. Fragezeichen (?)

Durch das Fragezeichen wird angegeben, dass die Rückgabe eines Argumentes mit dem angegebenen Name erwartet wird. Die Angabe ist nur in der Rückgabe-Argumentliste bei CMD bzw. EVT (nicht aber bei RET) erlaubt.

## 5. Rückgabe-Argumentliste (`<return-arg-list>`)

Die Rückgabe-Argumentliste ist optional und gibt an, welche Argumente als Antwort auf einen Befehl (CMD) oder ein Ereignis (EVT) erwartet werden oder bei der Antwort (RET) zurückgegeben wurden. Der Aufbau ist identisch zur Argumentliste.

## 2. Beispiele

### 1. Befehl

```
CMD Designer.Forms.Open (Name="MyFrame", ReadOnly=true) []
```

Hiermit wird der Befehl (CMD) Designer.Forms.Open zum Öffnen des Frames "MyFrame" im Read-Only-Modus definiert. Es werden keine Rückgabe-Argumente erwartet.

### 2. Ereignis

```
EVT Designer.Forms.CloseDone (Name="MyFrame", Type="0") []
```

Hiermit wird das Ereignis (EVT) Designer.Forms.CloseDone spezifiziert. Dieses Ereignis wird vom Designer ausgelöst, wenn eine Designform geschlossen wurde. Es wird auf keine Antwort des

## Kontakt

Plugin-Client gewartet (Rückgabe-Argumentliste ist leer).

### 3. Antwort

Die Plugin-Anwendung muss sich nach Aufbau der Verbindung beim Designer anmelden. Hierzu sendet der Designer folgendes Kommando: (1234 steht exemplarisch für die vom Designer generierte fortlaufende Antwortkennung.)

```
1234 | CMD Designer.Auth (Area="codelib", user="Alfons") [PluginName=?, Password=
```

Der Plugin-Client antwortet durch Rückgabe des entsprechenden Kommandos:

```
1234 | RET [PluginName="MyPlugin", Password="Password1234"]
```

## 2. Verbindungsherstellung

Nachdem der Designer gestartet wurde, lauscht dieser auf Verbindungen am spezifizierten Port des lokalen Rechners. Eine Plugin-Anwendung kann anschließend die Verbindung herstellen, indem diese eine Socketverbindung auf den Port des Designers herstellt.

### 3. Authentifikation

Nach Verbindungsherstellung sendet der Designer das Kommando Designer.Auth unter Angabe von Datenbankname sowie Datenbankbenutzer. Die Plugin-Anwendung muss innerhalb von 5 Sekunden auf dieses Kommando unter Angabe von Plugin-Name und Kennwort antworten.

Designer:

```
<serial> | CMD Designer.Auth (Area=<area-name>, User=<user-name>) [PluginName=?, Password=
```

Plugin-Anwendung:

```
<serial> | RET (PluginName=<plugin-name>, Password=<password>)
```

<serial> kennzeichnet die vom Designer automatisch vergebenene, fortlaufende Antwortkennung. Diese muss beim RET-Kommando wieder angegeben werden.

Bei erfolgreicher Anmeldung antwortet der Designer mit dem Kommando:

```
CMD Designer.Auth (Area=<area-name>, User=<user-name>, AccessGranted=true)
```

Bei nicht erfolgreicher Anmeldung wird AccessGranted=false zurückgesendet. Bei Erreichen des Timeouts von 5 Sekunden wird die Verbindung vom Designer geschlossen.

Das für die Plugin-Authentifikation notwendige Kennwort muss beim Benutzer in der Benutzerverwaltung (Seite Sicherheit) der Datenbank hinterlegt sein. Wenn das Kennwort leer ist, startet der Designer **ohne** Plugin-Funktionalität.



## Kontakt



Fehler, die bei der Authentifizierung auftreten, werden unter Angabe des PluginName im Anwendungsprotokoll des Designers festgehalten.

Plugin-Schnittstelle - Designer-Befehle

Befehle der Plugin-Schnittstelle für den Designer

Plugin-Schnittstelle,

Ereignisse,

Siehe Funktionsweise,

Plugin-Schnittstelle

in CONZEPT 16

Über die Plugin-Schnittstelle können Befehle an den Designer gesendet werden. Diese Befehle lösen im verbundenen Designer bestimmte Aktionen aus.

## Überblick

1. **Designer.Auth**
2. **Designer.GetInfo**
3. **Designer.Forms.\***
  1. **Designer.Forms.Open**
  2. **Designer.Forms.Export**
  3. **Designer.Forms.Import**
  4. **Designer.Forms.GetCount**
  5. **Designer.Forms.GetObject**
  6. **Designer.Forms.Selection.Get**
  7. **Designer.Forms.Selection.New**
  8. **Designer.Forms.Selection.GetCount**
  9. **Designer.Forms.Selection.GetObject**
  10. **Designer.Forms.Selection.GetPreview**
  11. **Designer.Forms.Selection.Menu.AddItem**
  12. **Designer.Forms.Selection.Menu.<Name>.Change**
  13. **Designer.Forms.Selection.Menu.<Name>.Remove**
4. **Designer.MenuEditor.Open**
5. **Designer.Editor.\***
  1. **Designer.Editor.Open**
  2. **Designer.Editor.Compile**
  3. **Designer.Editor.Document.Menu.AddItem**
  4. **Designer.Editor.Document.Menu.<Name>.Change**
  5. **Designer.Editor.Document.Menu.<Name>.Remove**
6. **Designer.Menu.\***
  1. **Designer.Menu.File.New.Frame.Exec**
  2. **Designer.Menu.File.New.AppFrame.Exec**
  3. **Designer.Menu.File.New.MdiFrame.Exec**
  4. **Designer.Menu.File.New.TrayFrame.Exec**
  5. **Designer.Menu.File.New.PrintDoc.Exec**
  6. **Designer.Menu.File.New.PrintDocRecord.Exec**
  7. **Designer.Menu.File.New.PrintForm.Exec**
  8. **Designer.Menu.File.New.PrintFormList.Exec**
  9. **Designer.Menu.File.New.Menu.Exec**
  10. **Designer.Menu.File.Open.Exec**
  11. **Designer.Menu.File.Reopen.Exec**
  12. **Designer.Menu.File.Save.Exec**
  13. **Designer.Menu.File.SaveAs.Exec**
  14. **Designer.Menu.File.ReadWrite.Exec**
  15. **Designer.Menu.File.Close.Exec**

16. **Designer.Menu.File.CloseAll.Exec**
  17. **Designer.Menu.File.Import.Exec**
  18. **Designer.Menu.File.Export.Exec**
  19. **Designer.Menu.File.Menu.Exec**
  20. **Designer.Menu.File.Test.Exec**
  21. **Designer.Menu.File.MainProc.Exec**
  22. **Designer.Menu.File.Config.Exec**
  23. **Designer.Menu.File.Exit.Exec**
  24. **Designer.Menu.Extras.Dictionary.Exec**
  25. **Designer.Menu.Extras.RecordManagement.Exec**
  26. **Designer.Menu.Extras.BlobManagement.Exec**
  27. **Designer.Menu.Extras.UserManagement.Exec**
  28. **Designer.Menu.Extras.Resource.Exec**
  29. **Designer.Menu.Extras.ViewAppLog.Exec**
  30. **Designer.Menu.Plugins.AddItem**
  31. **Designer.Menu.Plugins.<Name>.Change**
  32. **Designer.Menu.Plugins.<Name>.Remove**
  7. **Designer.Storage.\***
    1. **Designer.Storage.Create**
    2. **Designer.Storage.Load**
    3. **Designer.Storage.NewObject**
    4. **Designer.Storage.<StorageName>.Save**
    5. **Designer.Storage.<StorageName>.Unload**
    6. **Designer.Storage.<ObjectName>.DeleteObject**
    7. **Designer.Storage.<ObjectName>.RenameObject**
    8. **Designer.Storage.<ObjectName>.WriteKey**
    9. **Designer.Storage.<ObjectName>.ReadKey**
    10. **Designer.Storage.<ObjectName>.ReadNames**
    11. **Designer.Storage.LastLockingInfo**
    12. **Designer.Storage.<ObjectName>.ExportFile**
    13. **Designer.Storage.<ObjectName>.ExportStream**
    14. **Designer.Storage.ImportFile**
    15. **Designer.Storage.ImportStream**
  8. **Designer.Plugins.Shutdown**
  9. **Designer.LastXmlError**
- 

## Designer-Befehle

Es ist möglich auf die Antwort eines gesendeten Befehls zu warten (synchrone Ausführung). Dies ist die Voraussetzung, damit angeforderte Daten auch vom Designer an das Plugin zurückgesendet werden können. Für die Abfrage des Resultates der Ausführung kann das Argument ExecResult übergeben werden.

Nachdem die Antwort empfangen wurde, enthält ExecResult einen numerischen Wert, der entweder 0 ist (Ausführung erfolgreich) oder negativ (in Abhängigkeit vom Befehl). Existiert der Befehl nicht, wird \_ErrUnavailable zurückgegeben.

Folgende Befehle können gesendet werden:

### 1. Designer.Auth

## Kontakt

Dieser Befehl wird vom Designer gesendet um die Plugin-Anwendung aufzufordern, sich zu authentifizieren, nachdem diese sich mit dem Designer verbunden hat.

```
<serial> | CMD Designer.Auth (Area=<Datenbankname>, User=<Benutzername>) PluginName=?, Pas
```

Serial = Fortlaufende Nachrichtenkenung. Diese muss bei der Antwort angegeben werden, um sie zuzuordnen.  
Area = Name der Datenbank, die im Designer geöffnet ist.  
User = Name des Benutzers, der im Designer angemeldet ist.  
PluginName = Deskriptiver Name für das Plugin. Muss von der Plugin-Anwendung zurückgeliefert werden. Der Name darf nicht leer sein, sonst schlägt die Authentifikation fehl.  
Password = Plugin-Kennwort. Dieses muss beim Benutzer <Benutzername> in der Benutzerverwaltung hinterlegt werden.

Auf dieses Kommando muss vom Plugin-Client innerhalb von 5 Sekunden wie folgt geantwortet werden:

```
<serial> | RET [PluginName="<Plugin-Name>", Password="<Plugin-Kennwort>"]
```

Unabhängig davon, ob die Authentifizierung fehlschlägt oder nicht, sendet der Designer den Befehl

```
CMD Designer.Auth (Area=<Datenbankname>, User=<Benutzername>, AccessGranted=<true|false>)
```

Das Argument AccessGranted gibt Aufschluss darüber, ob die Anmeldung erfolgreich war oder nicht. Im letzteren Fall wird die Verbindung vom Designer terminiert.



Dieser Befehl wird vom Designer an das Plugin versendet.

## 2. Designer.GetInfo

Dieser Befehl liefert Informationen zum Designer. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.GetInfo () [Version=?, VersionNum=?, ExecResult=?]
```

Version = Designer-Version z. B. "5.8.01a"

VersionNum = Version als numerischer Wert um Versionsvergleiche einfach durchführen zu können (z. B. 0x05080101 = 05.08.01a).

## 3. Designer.Forms.\*

### 1. Designer.Forms.Open

Dieser Befehl öffnet eine Designform im Designer. Wenn sie bereits offen ist, wird sie aktiviert.

```
CMD Designer.Forms.Open (Name="<Designform-Name>", ReadOnly=<true|false>, Type=<Des
```

## Kontakt

Name = Name der zu öffnenden Designform.

ReadOnly = true, wenn die Designform im ReadOnly-Modus geöffnet werden soll, sonst false (optional; default = false).

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

### 2. Designer.Forms.Export

Dieser Befehl liefert ein im Designer geöffnetes Formular als Datenstrom in einem vorgegebenen Format. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.Export (FormName="<Form-Name>", Type=<Form-Typ>, FileName="<File
```

FormName = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

FileName = Pfad und Name der Datei, die erstellt werden soll.

FileFormat = Format der Datei (vs-resource (default) = vectorsoft resource format, xml = XML-Format).

### 3. Designer.Forms.Import

Dieser Befehl importiert eine Ressource im vectorsoft resource format und öffnet das Formular im Designer. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.Import (FileName="<File-Name>") [ExecResult=?]
```

FileName = Pfad und Name der Datei, die importiert und geöffnet werden soll.

### 4. Designer.Forms.GetCount

Dieser Befehl ermittelt die Anzahl der im Designer geöffneten Design-Forms. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.GetCount () [Count=?, ExecResult=?]
```

Count = Anzahl der geöffneten Design-Forms.

### 5. Designer.Forms.GetObject

Dieser Befehl ermittelt Informationen über eine im Designer geöffnete Design-Form. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.GetObject (FormName="<Form-Name>", Type=<Form-Typ>) [Name=?, Cla
```

FormName =

## Kontakt

Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type	= Typ der Designform (0 = <u>Frame</u> , 1 = <u>PrintForm</u> , 2 = <u>PrintFormList</u> , 3 = <u>PrintDoc</u> , 4 = <u>PrintDocRecord</u> ) (optional; default = 0).
Name	= Names der Design-Form.
Class	= Objekttyp (Wert einer <u>WinType</u> -Konstanten).
Caption	= Wert der Eigenschaft <u>Caption</u> der Design-Form als Base64-kodierte Zeichenkette.
State	= Status der Design-Form (new = neu angelegt und noch nicht gespeichert, read-only = nur-lesend geöffnet, modified = gespeichert und geändert, saved = gespeichert und nicht geändert).

### 6. Designer.Forms.Selection.Get

Dieser Befehl ermittelt das bzw. die im Designer selektierten Objekte als Base64 kodierte(n) Datenstrom. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.Selection.Get (FormName="<Form-Name>", Type=<Form-Typ>) [Content=
```

FormName = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

Content = Base64 kodierter Datenstrom der selektierten Objekte.

### 7. Designer.Forms.Selection.New

Dieser Befehl erstellt die, in einem zuvor mit Designer.Foms.Selection.Get ermittelten, Objekte im Designer und selektiert diese. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.Selection.New (FormName="<Form-Name>", Type=<Form-Typ>, Content=
```

FormName = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

Content = Base64 kodierter Datenstrom der selektierten Objekte.

### 8. Designer.Forms.Selection.GetCount

Dieser Befehl ermittelt die Anzahl der selektierten Objekte im Designer.

## Kontakt

Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Forms.Selection.GetCount (FormName="<Form-Name>", Type=<Form-Typ>) [Co

FormName = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

Count = Anzahl der selektierten Objekte.

### 9. Designer.Forms.Selection.GetObject

Dieser Befehl ermittelt Informationen über ein im Designer selektiertes Objekt. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Forms.Selection.GetObject (FormName="<Form-Name>", Type=<Form-Typ>, In

FormName = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

Index = Eins-basierter Index des gewünschten Objektes in der Selektion.

Name = Names des Objektes.

Class = Objekttyp (Wert einer WinType-Konstante).

Caption = Wert der Eigenschaft Caption des Objektes als Base64-kodierte Zeichenkette.

### 10. Designer.Forms.Selection.GetPreview

Dieser Befehl erstellt ein Vorschaubild des bzw. der selektierten Objekte und liefert die Daten im PNG-Format als Base64 kodierten Datenstrom. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

Das erstellte PNG hat eine maximale Größe von 256x256 Pixeln.

CMD Designer.Forms.Selection.GetPreview (FormName="<Form-Name>", Type=<Form-Typ>, I

FormName = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord) (optional; default = 0).

Content = Daten im PNG-Format als Base64 kodierter Datenstrom.

Width = Breite des Bildes in Pixel. (Ab Version 5.8.04; optional;

default = 0)

Height = Höhe des Bildes in Pixel. (Ab Version 5.8.04; optional; default = 0)

Ist Width oder Height auf -1 gesetzt, wird ein Vorschaubild erzeugt, welches die Größe des bzw. der selektierten Objekte besitzt. Ist andernfalls eines der beiden Argumente 0 oder nicht gesetzt, wird ein Vorschaubild mit 256 x 256 Pixel erzeugt.

## 11. Designer.Forms.Selection.Menu.AddItem

Dieser Befehl generiert einen Menüpunkt im Kontextmenü des selektierten bzw. der selektierten Objekte im Designer. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Forms.Selection.Menu.AddItem (Name="<Name>", Separator=<true|false>, C

Name = Name für den Menüpunkt. Dieser Name wird in Befehlen und Ereignissen verwendet, die den Menüpunkt betreffen.

Separator = Bei Angabe von true wird ein Separator erstellt. Bei false finden die weiteren Argumente Anwendung.

Caption = Anzeigetext für den Menüpunkt.

Icon = Name eines benutzerdefinierten Icons, dass vor dem Text angezeigt wird. Es kann sich um ein externes oder ein internes Icon handeln (siehe Icon).

ImageTile = Nummer eines vordefinierten CONZEPT 16 Symbols.

Checked = Bei Angabe von true wird der Menüpunkt mit einem Häkchen versehen.

Disabled = Bei Angabe von true wird der Menüpunkt inaktiv und ist nicht mehr anwählbar.



Menüpunkte werden in alphabetischer Reihenfolge des Namens im Menü hinzugefügt. Soll eine bestimmte Sortierung erreicht werden, kann mit einem Präfix gearbeitet werden.

Beispiel:

CMD Designer.Forms.Selection.Menu.AddItem (Name="Name3",Caption="Menüpunkt 1") [ ]CM

Die sichtbare Reihenfolge der Menüpunkte im Menü ist:

Menüpunkt 3

Menüpunkt 2

Menüpunkt 1

CMD Designer.Forms.Selection.Menu.AddItem (Name="01Name",Caption="Menüpunkt 1") [ ]C

Die sichtbare Reihenfolge der Menüpunkte im Menü ist:

Menüpunkt 1

Menüpunkt 2

Menüpunkt 3

## 12. Designer.Forms.Selection.Menu.<Name>.Change



## Kontakt

Mit diesem Befehl kann ein zuvor unter dem Namen <Name> angelegter Menüpunkt geändert werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.Selection.Menu.<Name>.Change (Separator=<true|false>, Caption="<
```

Name = Name des zu ändernden Menüpunktes.

Separator = Bei Angabe von true wird ein Separator erstellt. Bei false finden die weiteren Argumente Anwendung.

Caption = Anzeigetext für den Menüpunkt.

Icon = Name eines benutzerdefinierten Icons, dass vor dem Text angezeigt wird. Es kann sich um ein externes oder ein internes Icon handeln (siehe Icon).

ImageTile = Nummer eines vordefinierten CONZEPT 16 Symbols.

Checked = Bei Angabe von true wird der Menüpunkt mit einem Häkchen versehen.

Disabled = Bei Angabe von true wird der Menüpunkt inaktiv und ist nicht mehr anwählbar.

### 13. Designer.Forms.Selection.Menu.<Name>.Remove

Mit diesem Befehl kann ein zuvor unter dem Namen <Name> angelegter Menüpunkt wieder entfernt werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Forms.Selection.Menu.<Name>.Remove () [ExecResult=?]
```

Name = Name des zu löschenden Menüpunktes.

## 4. Designer.MenuEditor.Open

Dieser Befehl öffnet ein Menü im Menü-Editor.

```
CMD Designer.MenuEditor.Open (Name="<Menü-Name>", ReadOnly=<true|false>) [ExecResult=?]
```

Name = Name des zu öffnenden Menüs.

ReadOnly = true, wenn das Menü im ReadOnly-Modus geöffnet werden soll, sonst false (optional; default = false).

## 5. Designer.Editor.\*

### 1. Designer.Editor.Open

Dieser Befehl öffnet ein Dokument im Editor.

```
CMD Designer.Editor.Open (Name="<Dokument-Name>", ReadOnly=<true|false>, Type=<Doku
```

Name = Name des zu öffnenden Dokumentes.

ReadOnly = true, wenn das Dokument im ReadOnly-Modus geöffnet werden soll, sonst false (optional; default = false).

Type = Typ des Dokumentes (0 = Prozedur, 1 = Text) (optional; default = 0).

## 2. **Designer.Editor.Compile**

Dieser Befehl übersetzt eine Prozedur im Editor.

CMD Designer.Editor.Compile (Name="<Dokument-Name>") [ExecResult=?]

Name = Name des zu übersetzenden Dokumentes.

## 3. **Designer.Editor.Document.Menu.AddItem**

Dieser Befehl generiert einen Menüpunkt im Kontextmenü des aktiven Dokumentes im Editor. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Editor.Document.Menu.AddItem (Name="<Name>", Separator=<true|false>,

Name = Name für den Menüpunkt. Dieser Name wird in Befehlen und Ereignissen verwendet, die den Menüpunkt betreffen.

Separator = Bei Angabe von true wird ein Separator erstellt. Bei false finden die weiteren Argumente Anwendung.

Caption = Anzeigetext für den Menüpunkt.

Icon = Name eines benutzerdefinierten Icons, dass vor dem Text angezeigt wird. Es kann sich um ein externes oder ein internes Icon handeln (siehe Icon).

ImageTile = Nummer eines vordefinierten CONZEPT 16 Symbols.

Checked = Bei Angabe von true wird der Menüpunkt mit einem Häkchen versehen.

Disabled = Bei Angabe von true wird der Menüpunkt inaktiv und ist nicht mehr anwählbar.



Menüpunkte werden in alphabetischer Reihenfolge des Namens im Menü hinzugefügt. Soll eine bestimmte Sortierung erreicht werden, kann mit einem Präfix gearbeitet werden. (Beispiel siehe Designer.Forms.Selection.Menu.AddItem)

## 4. **Designer.Editor.Document.Menu.<Name>.Change**

Mit diesem Befehl kann ein zuvor unter dem Namen <Name> angelegter Menüpunkt geändert werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Editor.Document.Menu.<Name>.Change (Separator=<true|false>, Caption="<

Name = Name des zu ändernden Menüpunktes.

Separator = Bei Angabe von true wird ein Separator erstellt. Bei false finden die weiteren Argumente Anwendung.

Caption = Anzeigetext für den Menüpunkt.

Icon = Name eines benutzerdefinierten Icons, dass vor dem Text angezeigt wird. Es kann sich um ein externes oder ein internes Icon handeln (siehe Icon).

ImageTile = Nummer eines vordefinierten CONZEPT 16 Symbols.

Checked = Bei Angabe von true wird der Menüpunkt mit einem Häkchen versehen.

Disabled = Bei Angabe von true wird der Menüpunkt inaktiv und ist nicht mehr anwählbar.

#### 5. **Designer.Editor.Document.Menu.<Name>.Remove**

Mit diesem Befehl kann ein zuvor unter dem Namen <Name> angelegter Menüpunkt wieder entfernt werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Editor.Document.Menu.<Name>.Remove () [ExecResult=?]

Name = Name des zu löschenden Menüpunktes.

#### 6. **Designer.Menu.\***

##### 1. **Designer.Menu.File.New.Frame.Exec**

Führt den Menübefehl 'Datei / Neu / Frame' aus.

CMD Designer.Menu.File.New.Frame.Exec () [ExecResult=?]

##### 2. **Designer.Menu.File.New.AppFrame.Exec**

Führt den Menübefehl 'Datei / Neu / AppFrame' aus.

CMD Designer.Menu.File.New.AppFrame.Exec () [ExecResult=?]

##### 3. **Designer.Menu.File.New.MdiFrame.Exec**

Führt den Menübefehl 'Datei / Neu / MdiFrame' aus.

CMD Designer.Menu.File.New.MdiFrame.Exec () [ExecResult=?]

##### 4. **Designer.Menu.File.New.TrayFrame.Exec**

Führt den Menübefehl 'Datei / Neu / TrayFrame' aus.

CMD Designer.Menu.File.New.TrayFrame.Exec () [ExecResult=?]

##### 5. **Designer.Menu.File.New.PrintDoc.Exec**

Führt den Menübefehl 'Datei / Neu / PrintDoc' aus.

CMD Designer.Menu.File.New.PrintDoc.Exec () [ExecResult=?]

##### 6. **Designer.Menu.File.New.PrintDocRecord.Exec**

Führt den Menübefehl 'Datei / Neu / PrintDocRecord' aus.

CMD Designer.Menu.File.New.PrintDocRecord.Exec () [ExecResult=?]

##### 7. **Designer.Menu.File.New.PrintForm.Exec**

Führt den Menübefehl 'Datei / Neu / PrintForm' aus.

CMD Designer.Menu.File.New.PrintForm.Exec () [ExecResult=?]

##### 8. **Designer.Menu.File.New.PrintFormList.Exec**

## Kontakt

Führt den Menübefehl 'Datei / Neu / PrintFormList' aus.

CMD Designer.Menu.File.New.PrintFormList.Exec () [ExecResult=?]

### 9. **Designer.Menu.File.New.Menu.Exec**

Führt den Menübefehl 'Datei / Neu / Menü...' aus.

CMD Designer.Menu.File.New.Menu.Exec () [ExecResult=?]

### 10. **Designer.Menu.File.Open.Exec**

Führt den Menübefehl 'Datei / Öffnen' aus.

CMD Designer.Menu.File.Open.Exec () [ExecResult=?]

### 11. **Designer.Menu.File.Reopen.Exec**

Führt den Menübefehl 'Datei / Wiederherstellen' aus.

CMD Designer.Menu.File.Reopen.Exec () [ExecResult=?]

### 12. **Designer.Menu.File.Save.Exec**

Führt den Menübefehl 'Datei / Speichern' aus.

CMD Designer.Menu.File.Save.Exec () [ExecResult=?]

### 13. **Designer.Menu.File.SaveAs.Exec**

Führt den Menübefehl 'Datei / Speichern unter' aus.

CMD Designer.Menu.File.SaveAs.Exec () [ExecResult=?]

### 14. **Designer.Menu.File.ReadWrite.Exec**

Führt den Menübefehl 'Datei / Bearbeitungsmodus' aus.

CMD Designer.Menu.File.ReadWrite.Exec () [ExecResult=?]

### 15. **Designer.Menu.File.Close.Exec**

Führt den Menübefehl 'Datei / Schließen' aus.

CMD Designer.Menu.File.Close.Exec () [ExecResult=?]

### 16. **Designer.Menu.File.CloseAll.Exec**

Führt den Menübefehl 'Datei / Alle schließen' aus.

CMD Designer.Menu.File.CloseAll.Exec () [ExecResult=?]

### 17. **Designer.Menu.File.Import.Exec**

Führt den Menübefehl 'Datei / Importieren...' aus.

CMD Designer.Menu.File.Import.Exec () [ExecResult=?]

### 18. **Designer.Menu.File.Export.Exec**

Führt den Menübefehl 'Datei / Exportieren...' aus.

CMD Designer.Menu.File.Export.Exec () [ExecResult=?]

**19. Designer.Menu.File.Menu.Exec**

Führt den Menübefehl 'Datei / Menü editieren...' aus.

CMD Designer.Menu.File.Menu.Exec () [ExecResult=?]

**20. Designer.Menu.File.Test.Exec**

Führt den Menübefehl 'Datei / Test' aus.

CMD Designer.Menu.File.Test.Exec () [ExecResult=?]

**21. Designer.Menu.File.MainProc.Exec**

Führt den Menübefehl 'Datei / Start-Prozedur ausführen' aus.

CMD Designer.Menu.File.MainProc.Exec () [ExecResult=?]

**22. Designer.Menu.File.Config.Exec**

Führt den Menübefehl 'Datei / Einstellungen...' aus.

CMD Designer.Menu.File.Config.Exec () [ExecResult=?]

**23. Designer.Menu.File.Exit.Exec**

Führt den Menübefehl 'Datei / Beenden' aus.

CMD Designer.Menu.File.Exit.Exec () [ExecResult=?]

**24. Designer.Menu.Extras.Dictionary.Exec**

Führt den Menübefehl 'Extras / Datenstruktureditor...' aus.

CMD Designer.Menu.Extras.Dictionary.Exec () [ExecResult=?]

**25. Designer.Menu.Extras.RecordManagement.Exec**

Führt den Menübefehl 'Extras / Datensatzverwaltung...' aus.

CMD Designer.Menu.Extras.RecordManagement.Exec () [ExecResult=?]

**26. Designer.Menu.Extras.BlobManagement.Exec**

Führt den Menübefehl 'Extras / BLOb-Verwaltung...' aus.

CMD Designer.Menu.Extras.BlobManagement.Exec () [ExecResult=?]

**27. Designer.Menu.Extras.UserManagement.Exec**

Führt den Menübefehl 'Extras / Benutzerverwaltung...' aus.

CMD Designer.Menu.Extras.UserManagement.Exec () [ExecResult=?]

**28. Designer.Menu.Extras.Resource.Exec**

Führt den Menübefehl 'Extras / Ressourcenverwaltung...' aus.

CMD Designer.Menu.Extras.Resource.Exec () [ExecResult=?]

**29. Designer.Menu.Extras.ViewAppLog.Exec**

Führt den Menübefehl 'Extras / Anwendungsprotokoll öffnen' aus.

CMD Designer.Menu.Extras.ViewAppLog.Exec () [ExecResult=?]

### 30. **Designer.Menu.Plugins.AddItem**

Dieser Befehl generiert einen Menüpunkt im Plugin-Menü im Hauptmenü des Designers. Der Menüpunkt wird im Menü "Plugins" des Designers erstellt. Jedes Plugin erhält dort automatisch ein eigenes Untermenü mit dem Anzeigename der Plugin-Anwendung, der bei der Authentifizierung übergeben wird. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Menu.Plugins.AddItem (Name="<Name>", Separator=<true|false>, Caption="<Caption>")

**Name** = Name für den Menüpunkt. Dieser Name wird in Befehlen und Ereignissen verwendet, die den Menüpunkt betreffen.

**Separator** = Bei Angabe von true wird ein Separator erstellt. Bei false finden die weiteren Argumente Anwendung.

**Caption** = Anzeigetext für den Menüpunkt.

**Icon** = Name eines benutzerdefinierten Icons, dass vor dem Text angezeigt wird. Es kann sich um ein externes oder ein internes Icon handeln (siehe Icon).

**ImageTile** = Nummer eines vordefinierten CONZEPT 16 Symbols.

**Checked** = Bei Angabe von true wird der Menüpunkt mit einem Häkchen versehen.

**Disabled** = Bei Angabe von true wird der Menüpunkt inaktiv und ist nicht mehr anwählbar.



Menüpunkte werden in alphabetischer Reihenfolge des Namens im Menü hinzugefügt. Soll eine bestimmte Sortierung erreicht werden, kann mit einem Präfix gearbeitet werden. (Beispiel siehe Designer.Forms.Selection.Menu.AddItem)

### 31. **Designer.Menu.Plugins.<Name>.Change**

Mit diesem Befehl kann ein zuvor unter dem Namen <Name> angelegter Menüpunkt geändert werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Menu.Plugins.<Name>.Change (Separator=<true|false>, Caption="<Anzeige-Name>")

**Name** = Name des zu ändernden Menüpunktes.

**Separator** = Bei Angabe von true wird ein Separator erstellt. Bei false finden die weiteren Argumente Anwendung.

**Caption** = Anzeigetext für den Menüpunkt.

**Icon** = Name eines benutzerdefinierten Icons, dass vor dem Text angezeigt wird. Es kann sich um ein externes oder ein internes Icon handeln (siehe Icon).

**ImageTile** = Nummer eines vordefinierten CONZEPT 16 Symbols.

**Checked** = Bei Angabe von true wird der Menüpunkt mit einem Häkchen versehen.

**Disabled** =

## Kontakt

Bei Angabe von true wird der Menüpunkt inaktiv und ist nicht mehr anwählbar.

### 32. **Designer.Menu.Plugins.<Name>.Remove**

Mit diesem Befehl kann ein zuvor unter dem Namen <Name> angelegter Menüpunkt wieder entfernt werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Menu.Plugins.<Name>.Remove () [ExecResult=?]
```

Name = Name des zu löschenden Menüpunktes.

## 7. **Designer.Storage.\***

Die Storage-Befehle ermöglichen es einer Plugin-Anwendung Daten in einer Verzeichnisstruktur zu speichern und zu laden. Die Daten werden in einem speziell dafür vorgesehenen Bereich in der Datenbank auf Designer-Seite abgelegt.

Die oberste Ebene der Verzeichnisstruktur wird durch den Pfad Designer.Storage abgebildet. Plugin-Anwendungen können unter diesem Knoten einen Storage-Bereich anlegen, der anschließend über seinen Name referenzierbar ist.

### 1. **Designer.Storage.Create**

Dieser Befehl erstellt einen Storage-Bereich im Designer und sperrt diesen für die weitere Verwendung. Die Erstellung schlägt fehl, wenn der Bereich bereits von einem Plugin gesperrt wurde. Der Bereich sollte von der Plugin-Anwendung mit dem Befehl Cmd.Designer.Storage.Unload entladen und entsperrt werden, wenn der Bereich nicht mehr benötigt wird oder die Plugin-Anwendung beendet wird. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Storage.Create (Name="<Storage-Name>", Overwrite=<true|false>) [ExecRe
```

Name = Name des Storage-Bereiches. Erlaubt sind alle Zeichen beginnend mit Space (ASCII-Wert 32) außer "\*", "?", "\", ".".

Overwrite = Ist diese Argument true, wird der Bereich überschrieben sofern vorhanden. Default-Wert ist false.

### 2. **Designer.Storage.Load**

Dieser Befehl lädt einen Storage-Bereich exklusiv oder lesend. Zum Entladen und Entsperren des Bereiches kann der Befehl Cmd.Designer.Storage.Unload verwendet werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

```
CMD Designer.Storage.Load (Name="<Storage-Name>", Create=<true|false>, ReadOnly=<tr
```

Name = Name des Storage-Bereiches. Erlaubt sind alle Zeichen beginnend mit Space (ASCII-Wert 32) außer "\*", "?", "\", ".".

Create = Ist dieses Argument true, wird der Bereich erstellt, wenn er nicht vorhanden ist.

ReadOnly = Ist diese Argument true, wird der Bereich nur lesend geöffnet. Bei false wird der Bereich exklusiv geöffnet und kann von anderen Plugin-Anwendungen nur noch lesend geöffnet werden.

### 3. Designer.Storage.NewObject

Dieser Befehl erstellt ein neues Storage-Objekt. Das Storage-Objekt speichert Daten in der Form "Key=Value". Auf diese Weise können dem Storage-Objekt beliebige Inhalte zugewiesen, gespeichert und auch wieder gelesen werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.



Der Befehl kann nur durchgeführt werden, wenn der Bereich von der Plugin-Anwendung zuvor exklusiv gesperrt wurde.

CMD Designer.Storage.NewObject (Name="<Objekt-Name>") [ExecResult=?]

Name = Name des Storage-Objektes.

Der Name kann auch Unterverzeichnisse enthalten, die vom Befehl automatisch erstellt werden, sofern nicht vorhanden.

Beispiel:

CMD Designer.Storage.NewObject (Name="MyObjectRoot.MyObject") []

Erstellt im Storage-Bereich das Verzeichnis MyObjectRoot und darunter das Objekt MyObject.

### 4. Designer.Storage.<StorageName>.Save

Änderungen die durch Designer.Storage.NewObject, Designer.Storage.<ObjectName>.WriteKey, etc. am Storage-Bereich vorgenommen wurden, werden nicht automatisch in die Datenbank übertragen. Dieser Befehl überträgt den Storage-Bereich in die Datenbank. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Storage.<StorageName>.Save () [ExecResult=?]

StorageName = Name des zu speichernden Storage-Bereiches, der zuvor mit Designer.Storage.Create oder Designer.Storage.Load geöffnet wurde.

### 5. Designer.Storage.<StorageName>.Unload

Entlädt einen zuvor erstellten bzw. geladenen Storage-Bereich und entsperrt diesen. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Storage.<StorageName>.Unload () [ExecResult=?]

StorageName =



Name des zu entladenden Storage-Bereiches, der zuvor mit Designer.Storage.Create oder Designer.Storage.Load geöffnet wurde.

## 6. **Designer.Storage.<ObjectName>.DeleteObject**

Dieser Befehl löscht ein Storage-Objekt, welches zuvor mit Designer.Storage.NewObject angelegt wurde. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.



Für die Durchführung muss der Storage-Bereich exklusiv geöffnet worden sein.

CMD Designer.Storage.<ObjectName>.DeleteObject () [ExecResult=?]

ObjectName = Name des Storage-Objektes.  
Der Name kann auch Unterverzeichnisse enthalten.

Beispiel:

CMD Designer.Storage.MyObjects.MyObject.DeleteObject () []

Löscht das Storage-Objekt MyObject im Unterverzeichnis MyObjects.

## 7. **Designer.Storage.<ObjectName>.RenameObject**

Dieser Befehl löscht ein Storage-Objekt, welches zuvor mit Designer.Storage.NewObject angelegt wurde. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.



Für die Durchführung muss der Storage-Bereich exklusiv geöffnet worden sein.

CMD Designer.Storage.<ObjectName>.RenameObject (NameNew="<Name>") [ExecResult=?]

ObjectName = Name des umzubenennenden Storage-Objektes. Der Name kann auch Unterverzeichnisse enthalten.

NameNew = Neuer Name im selben Verzeichnis.

## 8. **Designer.Storage.<ObjectName>.WriteKey**

Dieser Befehl schreibt beliebige Inhalte zusammen mit deren Schlüssel in ein Storage-Objekt. Über die Schlüssel können die Werte mit dem Befehl Designer.Storage.<ObjectName>.ReadKey wieder ausgelesen werden. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.



Für die Durchführung muss der Storage-Bereich exklusiv geöffnet worden sein.

CMD Designer.Storage.<ObjectName>.WriteKey (<Key>=<Value>, <Key>=<Value>, <Key>=<Value>)

ObjectName = Name des Storage-Objektes. Der Name kann auch Unterverzeichnisse enthalten.

Key = Frei wählbarer Name, der mit <Value> assoziiert wird.  
Der Name kann auch Unterverzeichnisse enthalten.

Value = Beliebige Daten der zulässigen Type (Zeichenkette, numerischer oder logischer Wert).

Beispiel:

CMD Designer.Storage.MyObjects.MyObject.WriteKey (Vorname="Bill", Nachname="Gates",

#### 9. **Designer.Storage.<ObjectName>.ReadKey**

Dieser Befehl liefert zuvor mit Designer.Storage.<ObjectName>.WriteKey geschriebene Inhalte eines Storage-Objektes anhand der Schlüsselwerte. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Storage.<ObjectName>.ReadKey () [<Key>=?, <Key>=?, <Key>=?, ..., ExecR

ObjectName = Name des Storage-Objektes. Der Name kann auch Unterverzeichnisse enthalten.

Key = Schlüssel, dessen Inhalt gelesen werden soll.

#### 10. **Designer.Storage.<ObjectName>.ReadNames**

Dieser Befehl liest die Unterverzeichnisse von <ObjectName> und gibt sie in Form einer durch ? separierten Liste zurück. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Storage.<ObjectName>.ReadNames () [ExecResult=?]

ObjectName = Name des Storage-Objektes. Der Name kann auch Unterverzeichnisse enthalten.

#### 11. **Designer.Storage.LastLockingInfo**

Dieser Befehl ermittelt Informationen zum sperrenden Objekt. Die Informationen werden aktualisiert, wenn die Befehle Designer.Storage.Create bzw. Designer.Storage.Load ausgeführt werden, und der zu verwendende Bereich bereits gesperrt ist. Dieses Kommando steht ab CONZEPT 16-Version 5.8.04 zur Verfügung.

CMD Designer.Storage.LastLockingInfo () [LockingPluginName=?, LockingUserName=?, Lo

LockingPluginName = Name des Plugins, dass den Bereich im Zugriff hat (sofern auf derselben Maschine).

LockingUserName = Name des sperrenden Benutzers.

LockingSysName = Name des Rechners, auf dem das Plugin läuft.

LockingUserID = ID des sperrenden Benutzers.

#### 12. **Designer.Storage.<ObjectName>.ExportFile**

Der Befehl exportiert ein Storage-Objekt (samt evtl. vorhandener Unterobjekten) als XML-Dokument. Dieses Kommando steht ab CONZEPT 16-Version 5.8.04 zur Verfügung.

CMD Designer.Storage.<ObjectName>.ExportFile (FileName="<Dateiname>") [ExecResult=?]

ObjectName = Name des zu exportierenden Storage-Objektes. Der Name kann Unterverzeichnisse enthalten.

FileName = Vollständiger Pfad für die zu generierende XML-Datei.

### 13. **Designer.Storage.<ObjectName>.ExportStream**

Der Befehl exportiert ein Storage-Objekt (samt evtl. vorhandener Unterobjekte) als Base64 kodierten Datenstrom.

```
CMD Designer.Storage.<ObjectName>.ExportStream () [Content=?, ExecResult=?]
```

ObjectName = Name des zu exportierenden Storage-Objektes. Der Name kann Unterverzeichnisse enthalten.

Content = Daten im XML-Format als Base64 kodierter Datenstrom.

### 14. **Designer.Storage.<ObjectName>.ImportFile**

Der Befehl importiert eine XML-Datei als Storage-Objekt. Der Befehl kann nur durchgeführt werden, wenn der Storage-Bereich von der Plugin-Anwendung zuvor exklusiv gesperrt wurde.

```
CMD Designer.Storage.ImportFile (Name="<Storage Pfad>", NameNew="<Storage Name>", F
```

Name = Pfad für das zu importierende Storage-Objekt.

NameNew = Name für das Storage-Objekt. Fehlt dieses Argument oder ist es leer, wird der Name aus der XML-Datei ermittelt.  
(optional)

FileName = Vollständiger Pfad mit Dateiname der zu importierenden XML-Datei.

Tritt ein Fehler bei der XML-Verarbeitung auf, kann der Befehl Designer.LastXmlError verwendet werden, um weitere Informationen zu erhalten.

### 15. **Designer.Storage.<ObjectName>.ImportStream**

Der Befehl importiert XML aus einem Base64 kodierten Datenstrom. Der Befehl kann nur durchgeführt werden, wenn der Storage-Bereich von der Plugin-Anwendung zuvor exklusiv gesperrt wurde.

```
CMD Designer.Storage.ImportStream (Name="<Storage Pfad>", NameNew="<Storage Name>"
```

Name = Pfad für das zu importierende Storage-Objekt.

NameNew = Name für das Storage-Objekt. Fehlt dieses Argument oder ist es leer, wird der Name aus der XML-Datei ermittelt.  
(optional)

Content = Daten im XML-Format als Base64 kodierter Datenstrom.

Tritt ein Fehler bei der XML-Verarbeitung auf, kann der Befehl Designer.LastXmlError verwendet werden, um weitere Informationen zu erhalten.

## 8. **Designer.Plugins.Shutdown**

## Kontakt

Dieser Befehl entfernt die Plugin-Anwendung aus dem Designer. Bestehende Referenzen wie Menüpunkte werden entfernt. Geöffnete Storage-Bereiche werden geschlossen und entsperrt. Die Socket-Verbindung wird abgebaut. Dieses Kommando steht ab API-Version 1.1 zur Verfügung.

CMD Designer.Plugins.Shutdown () [ExecResult=?]

### 9. **Designer.LastXmlError**

Der Befehl liefert zusätzliche Informationen, wenn bei der XML-Verarbeitung durch Designer.Storage.ImportStream oder Designer.Storage.ImportFile ein Fehler auftrat. Dieses Kommando steht ab CONZEPT 16-Version 5.8.04 zur Verfügung.

CMD Designer.LastXmlError () [Error=?, ErrorText=?, ErrorLine=?, ErrorColumn=?, ErrorLevel=]

Error = XML-spezifischer Fehlercode.

ErrorText = XML-Fehlertext.

ErrorLine = Zeilennummer, in der der Fehler auftrat (sofern vorhanden, sonst 0).

ErrorColumn = Spalte, in der der Fehler auftrat (sofern vorhanden, sonst 0).

ErrorLevel = Fehlereinstufung ("none"=kein Fehler, "warning"=Warnung, "error"=Fehler, "fatal"=Fataler Fehler).

Plugin-Schnittstelle - Ereignisse  
Ereignisse der Plugin-Schnittstelle  
Plugin-Schnittstelle,  
Befehle,

Siehe Funktionsweise,  
Plugin-Schnittstelle  
in CONZEPT 16

Bei bestimmten Aktionen im Designer werden über die Plugin-Schnittstelle in den verbundenen Plugin-Clients Ereignisse ausgelöst.



Ist die Ursache der Aktion ein Befehl, welches über die Plugin-Schnittstelle versendet wurde, wird kein Ereignis ausgelöst.

## Überblick

1. **Designer.Forms.OpenDone**
2. **Designer.Forms.CloseDone**
3. **Designer.Forms.ActivateDone**
4. **Designer.Forms.SaveDone**
5. **Designer.Forms.WriteLockChangeDone**
6. **Designer.Forms.Selection.Menu.MenuSelect**
7. **Designer.Dictionary.Enter**
8. **Designer.Dictionary.Exit**
9. **Designer.Editor.OpenDone**
10. **Designer.Editor.CloseDone**
11. **Designer.Editor.ActivateDone**
12. **Designer.Editor.CompileDone**
13. **Designer.Editor.SaveDone**
14. **Designer.Editor.Document.Menu.MenuSelect**
15. **Designer.MenuEditor.OpenDone**
16. **Designer.MenuEditor.SaveDone**
17. **Designer.Menu.Plugins.MenuSelect**
18. **Designer.Exit**

## Ereignisse des Designers

Folgende Ereignisse werden vom Designer generiert:

### 1. **Designer.Forms.OpenDone**

Dieses Ereignis wird ausgelöst, nachdem eine Designform im Designer geöffnet wurde.

EVT Designer.Forms.OpenDone (Import=<true|false> Name="<Designform-Name>", ReadOnly=<true|

Import = true, wenn die Designform in den Designer importiert wurde, sonst false.

Name = Name der geöffneten Designform.

ReadOnly = true, wenn die Designform im ReadOnly-Modus geöffnet wurde, sonst false.

Type =

Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

## 2. Designer.Forms.CloseDone

Dieses Ereignis wird ausgelöst, nachdem eine Designform im Designer geschlossen wurde.

```
EVT Designer.Forms.CloseDone (Name="<Designform-Name>", Type=<Designform-Typ>) []
```

Name = Name der geöffneten Designform.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

## 3. Designer.Forms.ActivateDone

Dieses Ereignis wird ausgelöst, nachdem eine Designform im Designer aktiviert wurde.

```
EVT Designer.Forms.ActivateDone (Name="<Designform-Name>", Type=<Designform-Typ>) []
```

Name = Name der geöffneten Designform.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

## 4. Designer.Forms.SaveDone

Dieses Ereignis wird ausgelöst, nachdem eine Designform im Designer gespeichert wurde.

```
EVT Designer.Forms.SaveDone (NameOld="<Designform-Name alt>", NameNew="<Designform-Name ne
```

NameOld = Name vor dem Umbenennen (Save-As).

NameNew = Name nach dem Umbenennen (oder leere Zeichenkette, wenn die Designform nicht umbenannt wurde).

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

## 5. Designer.Forms.WriteLockChangeDone

Dieses Ereignis wird ausgelöst, wenn der Bearbeitungsmodus einer im Designer geöffneten Design-Form von nur-lesend auf schreibend oder umgekehrt verändert wurde.

```
EVT Designer.Forms.WriteLockChangeDone (Name="<Designform-Name>", Type=<Form-Typ>, ReadOnl
```

Name = Name des im Designer geöffneten Formulars. Fehlt das Argument oder ist es leer, wird das aktive Formular referenziert.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

ReadOnly = Bearbeitungsmodus (true = nur-lesend, false = lesend und schreibend).

## 6. Designer.Forms.Selection.Menu.MenuSelect

## Kontakt

Dieses Ereignis wird ausgelöst, wenn ein von der Plugin-Anwendung erstellter Menüpunkt im Kontextmenü der selektierten Objekte im Designer ausgewählt wurde.

EVT Designer.Forms.Selection.Menu.MenuSelect (FormName="<Form-Name>", Type=<Form-Typ>, Name=<Menüpunkt-Name>)

FormName = Name des im Designer geöffneten Formulars.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

Name = Name des ausgewählten Menüpunktes.

### 7. **Designer.Dictionary.Enter**

Dieses Ereignis wird ausgelöst, bevor der Datenstruktureditor angezeigt wird.

EVT Designer.Dictionary.Enter () []

### 8. **Designer.Dictionary.Exit**

Dieses Ereignis wird ausgelöst, nachdem der Datenstruktureditor geschlossen wurde.

EVT Designer.Dictionary.Exit () []

### 9. **Designer.Editor.OpenDone**

Dieses Ereignis wird ausgelöst, nachdem ein Dokument im Editor geöffnet wurde.

EVT Designer.Editor.OpenDone (Import=<true|false> Name="<Dokument-Name>", ReadOnly=<true|false> Type=<Dokument-Typ>)

Import = true, wenn das Dokument in den Editor importiert wurde, sonst false.

Name = Name des geöffneten Dokumentes.

ReadOnly = true, wenn das Dokument im ReadOnly-Modus geöffnet wurde, sonst false.

Type = Typ des Dokumentes (0 = Prozedur, 1 = Text).

### 10. **Designer.Editor.CloseDone**

Dieses Ereignis wird ausgelöst, nachdem ein Dokument im Editor geschlossen wurde.

EVT Designer.Editor.CloseDone (Name="<Dokument-Name>", Type=<Dokument-Typ>) []

Name = Name des geöffneten Dokumentes.

Type = Typ des Dokumentes (0 = Prozedur, 1 = Text).

### 11. **Designer.Editor.ActivateDone**

Dieses Ereignis wird ausgelöst, nachdem ein Dokument im Editor aktiviert wurde.

EVT Designer.Editor.ActivateDone (Name="<Dokument-Name>", Type=<Dokument-Typ>) []

Name = Name des geöffneten Dokumentes.

Type = Typ des Dokumentes (0 = Prozedur, 1 = Text).

## 12. **Designer.Editor.CompileDone**

Dieses Ereignis wird ausgelöst, nachdem eine Prozedur im Editor übersetzt wurde.

EVT Designer.Editor.CompileDone (Name="<Dokument-Name>", Error=<Fehlerwert>, ErrorLine=<Zeilennummer>)

Name = Name des geöffneten Dokumentes.

Error = Fehlerwert, wenn bei der Übersetzung ein Fehler auftrat.  
ErrOk, wenn die Übersetzung erfolgreich war.

ErrorLine = Zeilennummer > 0 im Fall einer fehlerhaften Übersetzung, sonst 0.

ErrorColumn = Spaltennummer > 0 im Fall einer fehlerhaften Übersetzung, sonst 0.

## 13. **Designer.Editor.SaveDone**

Dieses Ereignis wird ausgelöst, nachdem ein Dokument im Editor gespeichert wurde.

EVT Designer.Editor.SaveDone (NameOld="<Dokumentname alt>", NameNew="<Dokumentname neu>", Type=<Typ>)

NameOld = Name vor dem Umbenennen (Save-As).

NameNew = Name nach dem Umbenennen (oder leere Zeichenkette, wenn das Dokument nicht umbenannt wurde).

Type = Typ des Dokumentes (0 = Prozedur, 1 = Text).

## 14. **Designer.Editor.Document.Menu.MenuSelect**

Dieses Ereignis wird ausgelöst, wenn ein von der Plugin-Anwendung erstellter Menüpunkt im Kontextmenü des Editors ausgewählt wurde.

EVT Designer.Editor.Document.Menu.MenuSelect (DocumentName="<Dokument-Name>", Type=<Dokumenttyp>, MenuName="<Menüname>")

DocumentName = Name des im Designer geöffneten Dokumentes.

Type = Typ des Dokumentes (0 = Prozedur, 1 = Text).

Name = Name des ausgewählten Menüpunktes.

## 15. **Designer.MenuEditor.OpenDone**

Dieses Ereignis wird ausgelöst, nachdem ein Menü im Menü-Editor geöffnet wurde.

EVT Designer.MenuEditor.OpenDone (NameOpened="<Name des geöffneten Menüs>", NameClosed="<Name des geschlossenen Menüs>")

NameOpened = Name des geöffneten Menüs.

NameClosed = Name des dadurch geschlossenen vorher bearbeiteten Menüs  
(oder leere Zeichenkette, wenn kein Menü offen war).

ReadOnly =



## Kontakt

true, wenn das Menü zum Bearbeiten geöffnet wurde. false, wenn das Menü zur Anzeige geöffnet wurde.

Import = true, wenn das Menü importiert wurde, sonst false.

### 16. **Designer.MenuEditor.SaveDone**

Dieses Ereignis wird ausgelöst, nachdem ein Menü im Menü-Editor gespeichert wurde.

EVT Designer.MenuEditor.SaveDone (NameOld="<Menüname alt>", NameNew="<Menüname neu>") []

NameOld = Name vor dem Umbenennen (Save-As).

NameNew = Name nach dem Umbenennen (oder leere Zeichenkette, wenn das Menü nicht umbenannt wurde).

### 17. **Designer.Menu.Plugins.MenuSelect**

Dieses Ereignis wird ausgelöst, wenn ein von der Plugin-Anwendung erstellter Menüpunkt im Plugin-Menü des Designers ausgewählt wurde.

EVT Designer.Menu.Plugins.MenuSelect (FormName="<Form-Name>", Type=<Form-Typ>, Name="<Menu

FormName = Name des im Designer geöffneten Formulars.

Type = Typ der Designform (0 = Frame, 1 = PrintForm, 2 = PrintFormList, 3 = PrintDoc, 4 = PrintDocRecord).

Name = Name des ausgewählten Menüpunktes.

### 18. **Designer.Exit**

Das Ereignis wird ausgelöst, wenn der Designer beendet werden soll. Die Verbindung zu den Plugin-Anwendungen wird fünf Sekunden lang aufrechterhalten, damit die Plugins auf die Beendigung des Designers reagieren können. Danach wird die Verbindung terminiert und der Designer beendet.

EVT Designer.Exit () []

Plugin-Schnittstelle - CONZEPT 16

Erstellung eine Plugin-Anwendung in CONZEPT 16

Siehe Plugin-Schnittstelle

Um den Entwicklungsprozess einer Plugin-Anwendung in CONZEPT 16 zu vereinfachen wurde eine API entwickelt, welche grundlegende Funktionen für das Versenden und Kodieren/Dekodieren von Nachrichten zu Verfügung stellt.



Die API wurde in CONZEPT 16 entwickelt und ist Bestandteil der CodeLibrary.

## Überblick

### 1. Core-API

#### 1. Instanziierung

1. Erstellung einer Instanz
2. Schließen einer Instanz
3. Schließen aller Instanzen
4. Iterieren der Instanzen
5. Ermitteln des JobControl-Objektes
6. Senden einer Nachricht
7. Empfangen einer Nachricht
8. Eindeutige Antwortkennung generieren
9. Informationen zur API ermitteln
10. Ermitteln des zuletzt aufgetretenen Fehlers

### 2. Converter-API

1. Authentifizierungsaufforderung erwarten
2. Authentifizierungsaufforderung beantworten
3. Erstellen eines neuen PluginCommand-Handles
4. Löschen eines PluginCommand-Handles
5. Leeren eines PluginCommand-Handles
6. Name des Plugin-Kommandos ermitteln
7. Name eines Plugin-Kommandos überprüfen
8. Ermitteln des Kommando-Typs
9. Überprüfung auf einen bestimmten Kommando-Typ
10. Ermitteln der Antwortkennung (fortlaufende Nummer)
11. Ermitteln des zuletzt aufgetretenen Fehlers
12. Setzen von Argumenten
  1. String-Argument
  2. 32-bit vorzeichenbehaftetes ganzzahliges Argument
  3. Logisches Argument
  4. String-Argument als Memory-Objekt
  5. Rückgabe-Argument
  6. Argument für Resultat des Designer-Befehls
13. Ermitteln von Argumenten
  1. Argument nach Name ermitteln
  2. Argument nach Nummer ermitteln
  3. Ermitteln der Anzahl der Argumente
  4. Ermitteln der Länge eines Argumentes
  5. Argumentwert vom Typ TypeAlpha ermitteln
  6. Argumentwert eines Base64 kodierten Argumentes vom Typ TypeAlpha ermitteln
  7. Argumentwert vom Typ TypeInt ermitteln

8. Argumentwert vom Typ `TypeLogic` ermitteln
  9. String-Argument als Memory-Objekt ermitteln
  14. Kodieren / dekodieren von Kommandos
    1. Kodieren eines Plugin-Kommandos in das Nachrichtenformat
    2. Dekodieren eines Plugin-Kommandos aus dem Nachrichtenformat
    3. Empfangen und dekodieren einer Nachricht
    4. Kodieren und Senden einer Nachricht
- 

## 1. Core-API

Die Core-API stellt Basisfunktionen für das Senden und Empfangen von Befehlen und Ereignissen zur Verfügung und besteht aus den Modulen `Plugin.Core` sowie `Plugin.Core.Inc`. `Plugin.Core.Inc` deklariert alle Funktionen, die Bestandteil der Core-API sind. Das Modul `Plugin.Core` implementiert diese Funktionalität und sollte nicht verändert werden, da ansonsten eine Kompatibilität mit zukünftigen Versionen der API nicht gewährleistet werden kann.



Funktionen, die nicht Bestandteil der Core-API sind (also solche, die nur in `Plugin.Core` jedoch **nicht** in `Plugin.Core.Inc` definiert sind, sollten nicht aufgerufen werden, da sich die Implementierung in Zukunft ändern kann.

### 1. Instanziierung

Eine Plugin-Anwendung kann aus ein oder mehreren Instanzen bestehen, die getrennt voneinander fungieren. Jede Plugin-Instanz verfügt über eine eigene Verbindung zum Designer und einen eigenen Thread zum Empfang von Designer-Ereignissen bzw. Befehlen.

#### 1. Erstellung einer Instanz

Eine Plugin-Instanz wird über die Funktion `Plugin.Core:InstanceNew` erstellt:

```
declare Plugin.Core:InstanceNew( aPluginPort : word;    // Plugin port to c
```

Der Funktion muss beim Aufruf der Port übergeben werden, auf dem sich die Instanz mit dem Designer verbinden soll. Die Funktion wartet solange, bis die Verbindung hergestellt werden konnte oder das angegebene Timeout erreicht wurde. Im Argument `aFrame` kann ein Frame-Handle übergeben werden. Diesem werden über das Ereignis EvtJob neu eingetroffene Nachrichten signalisiert.



Die Angabe des Frame-Handles ist nur im Standard- bzw. Advanced-Client erlaubt. Bei Verwendung des SOA-Service muss hier NULL übergeben werden (alternativ kann das Argument auch ganz weggelassen werden).

## Kontakt

Bei erfolgreicher Verbindungsherstellung wird eine Instanz-ID (> 0) zurückgegeben, die für alle nachfolgenden Befehle der Core-API verwendet werden kann. Bei nicht erfolgreicher Verbindungsherstellung wird einer der folgenden Fehlercodes (< 0) zurückgeliefert.

<code>_ErrPluginCorePlatform</code>	Die Funktion wurde nicht vom Standard-, Advanced- oder SOA-Service aus aufgerufen.
<code>_ErrPluginCoreLimitReached</code>	Das Limit für Plugin-Instanzen (derzeit 10) ist bereits erreicht.
<code>_ErrPluginCoreConnectTimeout</code>	Es konnte keine Verbindung hergestellt werden (Timeout wurde erreicht).
<code>_ErrPluginCoreSckConnect</code>	Es konnte keine Verbindung hergestellt werden (Socket-Fehler / <u><code>Plugin.Core:GetLastError()</code></u> liefert den aufgetretenen Fehlercode).
<code>_ErrPluginCoreInternal</code>	Interner Fehler aufgetreten.

### 2. Schließen einer Instanz

Terminiert sich die Plugin-Anwendung oder wird die Instanz nicht mehr benötigt, dann sollte diese durch Aufruf der Funktion `Plugin.Core:InstanceClose` geschlossen werden.

```
declare Plugin.Core:InstanceClose( aID : int;           // Plugin instance ID
```

Im Argument `aID` muss die von `InstanceNew` zurückgelieferte Instanz-ID angegeben werden. Die Funktion baut die Verbindung zum Designer ab und terminiert den Empfangs-Thread. Die Funktion hat keinen Rückgabewert.

### 3. Schließen aller Instanzen

Zum Schließen aller Instanzen einer Plugin-Anwendung kann die Funktion `Plugin.Core:InstanceCloseAll` verwendet werden.

```
declare Plugin.Core:InstanceCloseAll();
```

### 4. Iterieren der Instanzen

Zum Iterieren der geöffneten Instanzen steht die Funktion `Plugin.Core:InstanceGet` zur Verfügung.

```
declare Plugin.Core:InstanceGet( aMode           : int; // _InstanceModeGetF
```

Im Argument `aMode` wird angegeben, was durchgeführt werden soll:

`_InstanceModeGetFirst` liefert die erste vorhandene Instanz zurück. Die Angabe von `aInstanceID` ist

## Kontakt

nicht notwendig.

`_InstanceModeGetNext` liefert ausgehend von `aInstanceID` die nächste vorhandene Instanz zurück.

Der Befehl liefert 0, wenn keine erste bzw. nächste Instanz existiert, ansonsten eine Instanz-ID (> 0).

### 5. Ermitteln des JobControl-Objektes

Zur Ermittlung des von einer Instanz verwendeten JobControl-Objektes kann die Funktion `Plugin.Core:GetJobControl` verwendet werden.

```
declare Plugin.Core:GetJobControl( aReceiverType : int; // Type of receiver
```

Das Argument `aReceiverType` kennzeichnet den Inhalt von `aReceiver`.

`_ReceiverByJobControl aReceiver` enthält das JobControl-Objekt.

`_ReceiverByInstanceID aReceiver` enthält die Instanz-ID.

Der Rückgabewert liefert im erfolgreichen Fall das JobControl-Objekt (> 0) oder einen Wert < 0 im Fehlerfall.

`_ErrPluginCoreInit` Es wurde noch kein InstanceNew() durchgeführt.

`_ErrPluginCoreNoInstance` Bei dem angegebenen Wert (`aReceiver`) handelt es sich nicht um eine gültige Instanz-ID.

### 6. Senden einer Nachricht

Zum Senden von Nachrichten steht die Funktion `Plugin.Core:SendLine` zur Verfügung.

```
declare Plugin.Core:SendLine( aInstanceID : int; // Instance ID. aMemReply
```

Die Funktion sendet eine Nachricht, deren Inhalt in einem Memory-Objekt enthalten ist. Das Nachrichtenformat ist hierbei wie unter (Nachrichtenaustausch) definiert.

Im Argument `aInstanceID` muss die Instanz-ID angegeben werden, welche die Nachricht versenden soll. Über das optionale Argument (`aWaitTimeout`) kann auf die Antwort einer Nachricht vom Designer gewartet werden. In diesem Fall muss auch das Argument (`aMemReply`) angegeben werden, welches den Inhalt der Antwort aufnimmt.



Das Versenden von Nachrichten über die Converter-API vereinfacht das Versenden nochmals, da hier ein zu sendender Befehl samt Argumente definiert werden kann.

Fehlerwerte:

## Kontakt

<u>_ErrPluginCoreInit</u>	Es wurde noch kein <u>InstanceNew()</u> durchgeführt.
<u>_ErrPluginCoreNoInstance</u>	Bei dem angegebenen Wert (aInstanceID) handelt es sich nicht um eine gültige Instanz-ID.
<u>_ErrPluginCoreSckWriteFailed</u>	Fehler beim Schreiben auf den Socket ( <u>Plugin.Core:GetLastError()</u> liefert den aufgetretenen Fehlercode).
<u>_ErrPluginCoreNoData</u>	Nur bei synchronem Versenden, wenn keine Daten vorhanden sind.
<u>_ErrData</u>	Unbekanntes MsxItem empfangen.
<u>_ErrPluginCoreThreadTerm</u>	Plugin-Thread hat sich beendet. Dies ist der Fall, wenn die Socket-Verbindung im <u>Designer</u> beendet wurde, z. B. weil der Designer vom Anwender beendet wurde (den genauen Fehlercode liefert <u>Plugin.Core:GetLastError()</u> ).
<u>_ErrPluginCoreReceive</u>	Fehler beim Empfangen der Daten, Fehlercode liefert <u>Plugin.Core:GetLastError()</u> .

## 7. Empfangen einer Nachricht

Es gibt zwei Arten, wie überprüft werden kann, ob eine neue Nachricht vorliegt:

### (a) Benachrichtigung durch Ereignis EvtJob.

Dies ist die bevorzugte Art im CONZEPT 16-Standard- oder Advanced-Client. Hierzu muss beim Aufruf von InstanceNew() ein Frame-Deskriptor angegeben werden.

### (b) Polling

Es kann in definierten Zeitabständen nach neuen Ereignissen "gepollt" werden. Diese Möglichkeit kann im SOA-Service genutzt werden.

Zum Empfangen und Auslesen von Daten steht die Funktion Plugin.Core:ReceiveLine() zur Verfügung.

```
declare Plugin.Core:ReceiveLine( aReceiverType : int; // Type of receiver
```

Der Funktion muss im ersten Argument eine der folgenden Konstanten angegeben werden:

\_ReceiverByJobControl Bei aReceiver handelt es sich um ein JobControl-Objekt. Dies ist sinnvoll, wenn

## Kontakt

der Aufruf der Funktion aus dem Ereignis EvtJob heraus stattfindet. Dem Ereignis wird bereits das JobControl-Objekt übergeben und kann somit direkt an diese Funktion weitergegeben werden.

\_ReceiverByInstanceID Bei aReceiver handelt es sich um eine Instanz-ID. Dies ist sinnvoll, wenn nach neuen Nachrichten gepollt werden soll.

Die Funktion liefert die empfangene Nachricht in aTargetMem zurück. Das Memory-Objekt muss zuvor mit MemAllocate(\_MemAutoSize) angelegt worden sein. Im Argument aInstanceID wird die übermittelte Instanz-ID zurückgegeben. Diese kann zu Verifikationszwecken herangezogen werden.

Das optionale Argument aWaitTimeout wird nur beim pollen verwendet, um eine gewisse Zeitspanne in Millisekunden auf das Eintreffen einer Nachricht zu warten. Im Ereignis EvtJob braucht das Argument nicht angegeben werden, da in diesem Fall bereits mindestens eine Nachricht empfangen wurde.

Mit dem optionalen Argument aWaitReply kann auf eine Antwort des Designers gewartet werden. Als aReceiverType muss \_ReceiverByInstanceID sein. Die Funktion wird intern von der Funktion Plugin.Core:SendLine() aufgerufen.

Rückgabewerte:

<u>_ErrOK</u>	Es wurde eine Nachricht in <u>aTargetMem</u> hinterlegt.
<u>_ErrPluginCoreInit</u>	Es wurde noch kein <u>InstanceNew()</u> durchgeführt.
<u>_ErrPluginCoreNoInstance</u>	Bei dem angegebenen Wert ( <u>aReceiver</u> ) handelt es sich nicht um eine gültige Instanz-ID.
<u>_ErrPluginCoreArgumentInvalid</u>	<u>aWaitReply</u> ist gesetzt und <u>aReceiverType</u> ist nicht <u>_ReceiverByInstanceID</u> , oder <u>aWaitReply</u> ist nicht gesetzt und <u>aReceiverType</u> wurde ein ungültiger Wert angegeben.
<u>_ErrPluginCoreNoData</u>	Es keine Daten vorhanden sind.
<u>_ErrPluginCoreThreadTerm</u>	Plugin-Thread hat sich beendet. Dies ist der Fall, wenn die Socket-Verbindung im <u>Designer</u> beendet wurde, z. B. weil der Designer vom Anwender beendet wurde (den genauen Fehlercode liefert <u>Plugin.Core:GetLastError()</u> ).

<code>_ErrPluginCoreReceive</code>	Fehler beim Empfangen der Daten, Fehlercode liefert <code>Plugin.Core:GetLastError()</code> .
------------------------------------	---

## 8. Eindeutige Antwortkennung generieren

Mit der Funktion `Plugin.Core:NextSerial()` kann eine eindeutige Antwortkennung generiert werden. Diese Funktion steht ab API-Version 1.1 zur Verfügung.

```
declare Plugin.Core:NextSerial(): bigint; // Next serial number
```

Im Fehlerfall liefert die Funktion den Wert `_ErrPluginCoreInit`. In diesem Fall wurde die Initialisierung der Core-API noch nicht durchgeführt.

## 9. Informationen zur API ermitteln

Mit der Funktion `Plugin.Core:ApiInfo()` können Informationen der API abgefragt werden. Bei Übergabe eines ungültigen Modus wird eine leere Zeichenkette zurückgeliefert.

```
declare Plugin.Core:ApiInfo( aMode : int; // Type of information)
```

Folgende Konstanten können in `aMode` angegeben werden.

<code>_ApiVersion</code>	Version der API als Zeichenkette. Derzeit wird hier '1.1.00' zurückgegeben.
<code>_ApiVersionCompare</code>	Es wird die Version als Zeichenkette zurückgegeben, die für eine einfachere Versionsprüfung verwendet werden kann. Derzeit wird hier '010100' zurückgegeben. (API-Ver 1.1)

## 10. Ermitteln des zuletzt aufgetretenen Fehlers

Zu einigen Fehlercodes können die nativen Fehlercodes ermittelt werden, die von den Funktionen der Core-API gesetzt werden. Hierzu dient die Funktion `Plugin.Core:GetLastError()`. Bei der Rückgabe von 0 ist kein Fehlercode vorhanden.

```
declare Plugin.Core:GetLastError(): int;
```

## 2. Converter-API

Zum Kodieren und Dekodieren von Nachrichten (wie unter Nachrichtenaustausch beschrieben) gibt es die Converter-API. Diese baut auf die Core-API auf und vereinfacht die Erstellung von Plugin-Befehlen und Ereignissen. Die API stellt einen abstrakten Datentyp `PluginCommand` bereit. Ein `PluginCommand` wird durch einen Deskriptor abgebildet.

Die Converter-API ist in den Modulen `Plugin.Converter` und `Plugin.Converter.Inc` enthalten.

### 1. Authentifizierungsaufforderung erwarten



## Kontakt

Die Funktion wartet auf die Nachricht `Designer.Auth`, die vom Designer zur Plugin-Anwendung gesendet wird, damit diese sich authentifiziert.

```
declare Plugin.Converter:ReceiveAuth( aReceiverType : int; // (in) Type of
```

Die Argumente `aReceiverType`, `aReceiver` und `aWaitTimeout` haben dieselbe Bedeutung wie unter Plugin.Core:ReceiveLine beschrieben. Liefert die Funktion ErrOk, wird in `aSerial` die vom Designer erwartete Antwortkennung zurückgegeben. In `aUser` wird der Name des Benutzers zurückgegeben, für den die Authentifizierung erfolgen soll.

### 2. Authentifizierungsaufforderung beantworten

Die Funktion sendet die Antwort auf die Authentifizierungsaufforderung, die durch die Funktion Plugin.Converter:ReceiveAuth empfangen wurde.

```
declare Plugin.Converter:ReplyAuth( aInstanceID : int; // Instanc
```

In `aSerial` wird die empfangene Antwortkennung übergeben.

In `aPluginName` wird ein Name für die Plugin-Anwendung übergeben. Der Name wird für Einträge, die das Plugin betreffen, im Anwendungslog des Designers verwendet. Der Name muss mindestens ein Zeichen und darf maximal 40 Zeichen umfassen und kann keine Steuerzeichen (< ASCII 32) sowie '?' und '\' enthalten. Er darf auch nicht nur aus Leerzeichen bestehen.

In `aPassword` muss das zum Benutzer gehörende Plugin-Kennwort übergeben werden.

In `aWaitTimeout` wird die maximale Zeitspanne in Millisekunden übergeben, die gewartet wird.

`aDescriptiveName` enthält einen Anzeigenamen für die Plugin-Anwendung. Dieser Name wird im Plugin-Menü des Designers angezeigt, wenn die Anwendung dort einen Menüeintrag erstellt.

`aDescription` enthält eine kurze Beschreibung der Plugin-Anwendung. Das Argument wird z. Zt. nicht ausgewertet.

Rückgabewert:

ErrOk Authentifizierung erfolgreich.

ErrRights Authentifizierung fehlgeschlagen.

### 3. Erstellen eines neuen PluginCommand-Handles

Zum Erstellen eines Plugin-Kommandos steht die Funktion `Plugin.Converter:CreateCmd()` zur Verfügung.

```
declare Plugin.Converter:CreateCmd( opt aKind : int; opt aName
```

Im ersten Argument wird die Art des Kommandos übergeben:

sPluginCmdKindEvt Es soll ein Ereignis generiert werden (EVT).  
sPluginCmdKindCmd Es soll ein Befehl generiert werden (CMD).  
sPluginCmdKindRet Es soll eine Antwort generiert werden (RET).  
Im optionalen Argument aName wird der Name des Kommandos angegeben. Bei der Angabe von sPluginCmdKindRet sollte das Argument nicht angegeben werden, da das RET-Kommando keinen Namen besitzt und sich ausschließlich auf die fortlaufende Nummer (aSerial) bezieht.

Mit der Funktion Plugin.Core:NextSerial kann eine eindeutige Antwortkennung für aSerial generiert werden.

Die Funktion liefert einen Deskriptor auf ein PluginCommand zurück. Der Deskriptor muss mit Plugin.Converter>DeleteCmd() freigegeben werden, wenn er nicht mehr benötigt wird.

#### 4. Löschen eines PluginCommand-Handles

```
declare Plugin.Converter>DeleteCmd( aPluginCmd      : handle;           // Handle o
```

Die Funktion löscht ein durch Plugin.Convert>CreateCmd() erstelltes Kommando. In aPluginCmd muss das zu löschende Handle angegeben werden.

#### 5. Leeren eines PluginCommand-Handles

Mit der Funktion Plugin.Converter:ClearCmd() kann ein PluginCommand-Handle für die Wiederverwendung geleert bzw. neu definiert werden.

```
declare Plugin.Converter:ClearCmd( aPluginCmd      : handle;           // Handle o
```

In aPluginCmd muss ein Handle auf ein mit Plugin.Converter>CreateCmd() generiertes PluginCommand übergeben werden.

#### 6. Name des Plugin-Kommandos ermitteln

Der Name des Plugin-Kommandos kann mit der Funktion Plugin.Convert:GetCmdName() ermittelt werden.

```
declare Plugin.Converter:GetCmdName( aPluginCmd      : handle;           // Handle o
```

#### 7. Name eines Plugin-Kommandos überprüfen

Mit der Funktion Plugin.Converter:IsCmdName() kann der Name eines Plugin-Kommandos auf Gleichheit (ohne Berücksichtigung von Groß-/Kleinschreibung) geprüft werden.

```
declare Plugin.Converter:IsCmdName( aPluginCmd      : handle;           // Handle o
```

#### 8. Ermitteln des Kommando-Typs

Mit der Funktion Plugin.Converter:GetCmdKind() kann der Typ eines Kommandos ermittelt werden.

```
declare Plugin.Converter:GetCmdKind( aPluginCmd      : handle;           // Handle o
```

Rückgabewert:

sPluginCmdKindEvt Ereignis

sPluginCmdKindCmd Befehl

sPluginCmdKindRet Antwort

## 9. Überprüfung auf einen bestimmten Kommando-Typ

Die folgenden Funktionen stehen für die Überprüfung auf einen bestimmten Kommando-Typ zur Verfügung:

```
declare Plugin.Converter:IsCmdKindEvt( aPluginCmd      : handle;           // Handl
```

```
declare Plugin.Converter:IsCmdKindCmd( aPluginCmd      : handle;           // Handl
```

```
declare Plugin.Converter:IsCmdKindRet( aPluginCmd      : handle;           // Handl
```

## 10. Ermitteln der Antwortkennung (fortlaufende Nummer)

Mit Plugin.Converter:GetCmdSerial() kann die im Kommando gesetzte Antwortkennung ermittelt werden. Ist der Rückgabewert 0, dann besitzt das Kommando keine Antwortkennung.

```
declare Plugin.Converter:GetCmdSerial( aPluginCmd      : handle;           // Handl
```

## 11. Ermitteln des zuletzt aufgetretenen Fehlers

Zu einigen Fehlercodes können die nativen Fehlercodes ermittelt werden, die von den Funktionen der Converter-API gesetzt werden. Hierzu dient die Funktion Plugin.Converter:GetLastError(). Bei der Rückgabe von 0 ist kein Fehlercode vorhanden.

```
declare Plugin.Converter:GetLastError( aPluginCmd      : handle;           // Handl
```

## 12. Setzen von Argumenten

Zum Setzen von Argumenten und deren zugehörigen Werten gibt es je nach Datentyp unterschiedliche Funktionen.

### 1. String-Argument

Mit der Funktion Plugin.Converter:AddArgStr() wird einem Plugin-Kommando ein String-Argument hinzugefügt.

```
declare Plugin.Converter:AddArgStr( aPluginCmd      : handle;           // H
```

Der Name des Argumentes muss im Parameter aName übergeben werden. Sein Wert wird im Argument aValue übergeben. aValueCharset gibt den Zeichensatz an, in dem aValue kodiert ist:

sPluginArgStrC16 CONZEPT 16-Zeichensatz

sPluginArgStrISO ANSI-Zeichensatz

sPluginArgStrUtf8 UTF-8 -Zeichensatz

Über das optionale Argument aReturnArg kann angegeben werden, ob das Argument in die Liste der Rückgabe-Argumente (true)

## Kontakt

eingefügt werden soll oder nicht (false). Letzteres ist der Default-Fall.

### 2. 32-bit vorzeichenbehaftetes ganzzahliges Argument

```
declare Plugin.Converter:AddArgInt( aPluginCmd      : handle;           // H
```

Analog zur Funktion AddArgStr(), nur dass ein Argument vom Type TypeInt generiert wird.

### 3. Logisches Argument

```
declare Plugin.Converter:AddArgLogic( aPluginCmd      : handle;           //
```

Analog zur Funktion AddArgStr(), nur dass ein Argument vom Type TypeLogic generiert wird.

### 4. String-Argument als Memory-Objekt

Diese Funktion steht ab API-Version 1.1 zur Verfügung.

```
declare Plugin.Converter:AddArgMem( aPluginCmd      : handle;           //
```

Die Funktion nimmt in aValue den Deskriptor eines Memory-Objektes entgegen. Der Inhalt des Memory-Objektes wird kopiert. Binäre Inhalte müssen konvertiert werden, so dass keine Steuerzeichen etc. enthalten sind. Hierfür gibt es die Option aMemCnvMode mit der eine der Optionen des CONZEPT 16-Befehls MemCnv() angegeben werden kann. Durch Angabe von MemEncBase64 wird der Inhalt des kopierten Memory-Objektes nach Base-64 kodiert und somit übertragen werden kann.

### 5. Rückgabe-Argument

Die Funktion Plugin.Converter:AddArgRet() generiert ein Rückgabe-Argument in der Form <name>=?.

```
declare Plugin.Converter:AddArgRet( aPluginCmd      : handle;           //
```

### 6. Argument für Resultat des Designer-Befehls

Die Funktion Plugin.Converter:AddExecResult() fügt dem Kommando ein Argument ExecResult hinzu. Dieses wird bei der Rückantwort vom Designer gesetzt und gibt Aufschluss über Fehler bei der Ausführung eines Designer-Befehls. Diese Funktion steht ab API-Version 1.1 zur Verfügung.

```
declare Plugin.Converter:AddExecResult( aPluginCmd      : handle;
```

## 13. Ermitteln von Argumenten

Analog zum Setzen von Argumenten werden Funktionen für das Ermitteln von Argumenten und Argumentwerte definiert.

### 1. Argument nach Name ermitteln

Die Funktion Plugin.Converter:GetArg() ermittelt ein Argument eines Plugin-Kommandos anhand seines Namens.

## Kontakt

```
declare Plugin.Converter:GetArg( aPluginCmd      : handle;           // Hand
```

Es wird ein Deskriptor auf das im Kommando gespeicherte CteNode-Objekt zurückgegeben. Der Typ des Arguments kann anhand der Eigenschaft Type bestimmt und danach mit der entsprechenden Eigenschaft Value... ausgelesen werden.

### 2. **Argument nach Nummer ermitteln**

Die Funktion Plugin.Converter:GetArgByNum() ermittelt ein Argument anhand seiner Position (1..Anzahl der Argumente). Die Position muss im Argument aArgNum übergeben werden.

```
declare Plugin.Converter:GetArgByNum( aPluginCmd      : handle;           //
```

### 3. **Ermitteln der Anzahl der Argumente**

Mit Plugin.Converter:GetArgCount() kann die Anzahl der gesetzten Argumente ermittelt werden.

```
declare Plugin.Converter:GetArgCount( aPluginCmd      : handle;           //
```

### 4. **Ermitteln der Länge eines Argumentes**

Mit Plugin.Converter:GetArgStrLen() kann die Länge eines gesetzten Argumentes vom Typ TypeAlpha ermittelt werden. Diese Funktion steht ab API-Version 1.1 zur Verfügung.

```
declare Plugin.Converter:GetArgStrLen( aPluginCmd      : handle;           /
```

Rückgabewerte:

>= 0                    Länge des Inhaltes. Ist das Argument nicht vom Typ TypeAlpha, wird 0 zurückgegeben.

ErrUnavailable Das Argument mit dem Name aName ist im Kommando aPluginCmd nicht vorhanden.

### 5. **Argumentwert vom Typ TypeAlpha ermitteln**

Die Funktion Plugin.Converter:GetArgStr() ermittelt den Wert eines Argumentes vom Typ TypeAlpha. Falls das angegebene Argument (aName) nicht vom Typ TypeAlpha ist, wird der Fehlercode ErrType zurückgegeben.

```
declare Plugin.Converter:GetArgStr( aPluginCmd      : handle;           // H
```

Rückgabewerte:

ErrOK                    Durchführung erfolgreich.

ErrUnavailable Das Argument mit dem Name aName ist im Kommando aPluginCmd nicht vorhanden.

ErrType                    Das Argument aName ist nicht vom Typ TypeAlpha.

### 6. **Argumentwert eines Base64 kodierten Argumentes vom Typ TypeAlpha ermitteln**

## Kontakt

Die Funktion `Plugin.Converter:GetArgStrDecB64()` ermittelt den Wert eines Argumentes vom Typ `_TypeAlpha`. Falls das angegebene Argument (`aName`) nicht vom Typ `_TypeAlpha` ist, wird der Fehlercode `_ErrType` zurückgegeben. Diese Funktion steht ab API-Version 1.1 zur Verfügung.

```
declare Plugin.Converter:GetArgStrDecB64( aPluginCmd      : handle;
```

Rückgabewerte:

<u><code>_ErrOK</code></u>	Durchführung erfolgreich.
<u><code>_ErrUnavailable</code></u>	Das Argument mit dem Name <code>aName</code> ist im Kommando <code>aPluginCmd</code> nicht vorhanden.
<u><code>_ErrType</code></u>	Das Argument <code>aName</code> ist nicht vom Typ <u><code>_TypeAlpha</code></u> .
<u><code>_ErrOutOfMemory</code></u>	Speicheranforderungsfehler.

### 7. Argumentwert vom Typ `_TypeInt` ermitteln

Die Funktion `Plugin.Converter:GetArgInt()` ermittelt den Wert eines Argumentes vom Typ `_TypeInt`. Falls das angegebene Argument (`aName`) nicht vom Typ `_TypeInt` ist, wird der Fehlercode `_ErrType` zurückgegeben.

```
declare Plugin.Converter:GetArgInt( aPluginCmd      : handle;           //
```

Rückgabewerte:

<u><code>_ErrOK</code></u>	Durchführung erfolgreich.
<u><code>_ErrUnavailable</code></u>	Das Argument mit dem Name <code>aName</code> ist im Kommando <code>aPluginCmd</code> nicht vorhanden.
<u><code>_ErrType</code></u>	Das Argument <code>aName</code> ist nicht vom Typ <u><code>_TypeInt</code></u> .

### 8. Argumentwert vom Typ `_TypeLogic` ermitteln

Die Funktion `Plugin.Converter:GetArgLogic()` ermittelt den Wert eines Argumentes vom Typ `_TypeLogic`. Falls das angegebene Argument (`aName`) nicht vom Typ `_TypeLogic` ist, wird der Fehlercode `_ErrType` zurückgegeben.

```
declare Plugin.Converter:GetArgLogic( aPluginCmd      : handle;           //
```

Rückgabewerte:

<u><code>_ErrOK</code></u>	Durchführung erfolgreich.
<u><code>_ErrUnavailable</code></u>	Das Argument mit dem Name <code>aName</code> ist im Kommando <code>aPluginCmd</code> nicht vorhanden.
<u><code>_ErrType</code></u>	Das Argument <code>aName</code> ist nicht vom Typ <u><code>_TypeLogic</code></u> .

### 9. String-Argument als Memory-Objekt ermitteln

## Kontakt

Die Funktion `Plugin.Converter:GetArgMem()` ermittelt den Wert eines Argumentes vom Typ TypeAlpha. Falls das angegebene Argument (`aName`) nicht vom Typ TypeAlpha ist, wird der Fehlercode ErrType zurückgegeben. Diese Funktion steht ab API-Version 1.1 zur Verfügung.

```
declare Plugin.Converter:GetArgMem( aPluginCmd      : handle;          // H
```

In `aValue` muss der Deskriptor eines Memory-Objektes angegeben werden, in welches der Inhalt des Argumentes kopiert wird. Das optionale Argument `aMemCnvMode` kann eine, der bei MemCnv() zulässigen, Optionen sein. Der Inhalt einer Base64 kodierten Zeichenkette kann z. B. durch Angabe der Option MemDecBase64 ermittelt werden.

Rückgabewerte:

<u>ErrOK</u>	Durchführung erfolgreich.
<u>ErrUnavailable</u>	Das Argument mit dem Name <code>aName</code> ist im Kommando <code>aPluginCmd</code> nicht vorhanden.
<u>ErrHdlInvalid</u>	Bei <code>aValue</code> handelt es sich nicht um einen gültigen Deskriptor eines Memory-Objektes.
<u>ErrType</u>	Das Argument <code>aName</code> ist nicht vom Typ <u>TypeAlpha</u> .
<u>ErrOutOfMemory</u>	Speicheranforderungsfehler.

## 14. Kodieren / dekodieren von Kommandos

### 1. Kodieren eines Plugin-Kommandos in das Nachrichtenformat

Um eine Nachricht an den Designer zu senden, muss das `PluginCommand-Handle` zunächst in das Nachrichtenformat (siehe Nachrichtenaustausch) umgewandelt (kodiert) werden. Zum Kodieren steht die Funktion `Plugin.Converter:Encode()` bereit.

```
declare Plugin.Converter:Encode( aPluginCmd      : handle;          // (in)
```

Im Argument `aPluginCmd` wird das zu kodierende `PluginCommand-Handle` übergeben. Im Argument `aOutput` muss ein Memory-Objekt übergeben werden, welches den kodierten Nachrichteninhalte aufnimmt. Das Memory-Objekt muss zuvor mit MemAllocate( MemAutoSize) generiert worden sein.

### 2. Dekodieren eines Plugin-Kommandos aus dem Nachrichtenformat

Zum Umwandeln einer Nachricht vom Designer in ein `PluginCommand-Handle` steht die Funktion `Plugin.Converter:Decode()` bereit. Diese nimmt eine Nachricht (die z. B. durch Plugin.Core:ReceiveLine() ermittelt wurde) und dekodiert diese in ein `PluginCommand-Handle`.

## Kontakt

```
declare Plugin.Converter:Decode( aInput          : handle;          // (in)
```

aPluginCmd muss ein PluginCommand-Handle sein, dass zuvor mit Plugin.Converter:CreateCmd() angelegt wurde. Bei erfolgreicher Verarbeitung wird der Wert \_ErrOk zurückgegeben, ansonsten ein Fehlercode < 0.

### 3. Empfangen und dekodieren einer Nachricht

Mit der Funktion Plugin.Converter:ReceiveCmd() kann ein Plugin-Kommando empfangen werden. Die Funktion ruft hierzu Plugin.Core:ReceiveLine() und anschließend Plugin.Converter:Decode() auf.

```
declare Plugin.Converter:ReceiveCmd( aReceiverType : int;          //
```

Die Argumente aReceiverType, aReceiver, aInstanceID und aWaitTimeout haben dieselbe Bedeutung wie unter Plugin.Core:ReceiveLine() beschrieben. Das Plugin-Kommando aPluginCmd muss zuvor mit Plugin.Converter:CreateCmd() generiert werden und wird von der Funktion entsprechend der empfangenen Nachricht dekodiert.

### 4. Kodieren und Senden einer Nachricht

Die Funktion Plugin.Converter:SendCmd() wandelt ein PluginCommand-Handle in eine Nachricht um und sendet diese zum Designer.

```
declare Plugin.Converter:SendCmd( aInstanceID      : int;          // (in)
```

Bei erfolgreicher Verarbeitung wird \_ErrOk zurückgegeben, ansonsten ein Wert < 0.



## FAQ

Häufig gestellte Fragen und deren Antworten

In diesem Abschnitt befinden sich häufig gestellte Fragen. Die Fragen sind nach folgenden Themen sortiert:

- **Client** - Allgemeine Fragen zum Client.
- **Server** - Allgemeine Fragen zum Server. Spezielle Sachverhalte zu den einzelnen Plattformen werden in weiteren Unterabschnitten erläutert.
- **Web-Schnittstelle** - Fragen zur Web-Schnittstelle, Einrichten und Programmierung.
- **Externe Windows Programmierschnittstelle** - Fragen zur Programmierschnittstelle.
- **ODBC-Schnittstelle** - Optimierung von Abfragen, Fehlermeldungen, ...
- **Installation** - Fragen zur Installation der unterschiedlichen Lizenzen.
- **Internetbasierte Lizenzen** - Fragen zu internetbasierten Lizenzen.
- **Listen** - Fragen zu Listen in CONZEPT 16.
- **Programmierung** - Fragen aus den unterschiedlichen Bereichen der Programmierung. Fertige Programmteile zum kopieren.
- **Verschiedenes** - Fragen zum Umfeld, Versionsumstieg, ...

## FAQ - Client

### Häufige Fragen zum Client

Warum kann die CONZEPT 16-Hilfe nicht angezeigt werden?

Wenn der Client über eine Netzwerkfreigabe aufgerufen wird, kann es sein, dass die Anzeige der Hilfe (c16.chm) durch Sicherheitsrichtlinien von Windows verhindert wird. In diesem Fall erscheint folgender Hinweis:

"Die Navigation zu der Webseite wurde abgebrochen."

Diese Sicherheitsrichtlinie soll verhindern, dass bösartiger Programm-Code in .chm-Dateien ausgeführt wird. Um die Hilfe dennoch anzeigen zu können, gibt es folgende Lösungsmöglichkeiten:

#### **1. Anzeige aller .chm-Dateien in bestimmten Netzwerkfreigaben erlauben (empfohlen)**

Bei dem Computer bei dem die Anzeige erlaubt werden soll muss die Registrierung angepasst werden. Unter dem Schlüssel

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\HTMLHelp\1.x\ItssRestrictions]
```

für 32-Bit-Systeme bzw.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\HTMLHelp\1.x\ItssRestrictions]
```

für 64-Bit-Systeme muss der Wert

UrlAllowList

als Zeichenfolge angelegt werden, falls er noch nicht vorhanden ist. Der Wert muss auf

```
\\<Servername>\<Freigabename>;file://\\<Servername>\<Freigabename>
```

gesetzt bzw. erweitert werden, um die Anzeige von .chm-Dateien in einer Netzwerkfreigabe zu erlauben. Mehrere Netzwerkfreigaben werden durch ein Semikolon (";") getrennt.

Die Anpassung betrifft alle Benutzerkonten des Computers.

#### **2. Anzeige aller .chm-Dateien im Intranet erlauben**

Bei dem Computer bei dem die Anzeige erlaubt werden soll muss die Registrierung angepasst werden. Unter dem Schlüssel

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\HTMLHelp\1.x\ItssRestrictions]
```

für 32-Bit-Systeme bzw.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\HTMLHelp\1.x\ItssRestrictions]
```

für 64-Bit-Systeme muss der Wert

MaxAllowedZone

als DWORD-Wert angelegt werden, falls er noch nicht vorhanden ist. Der Wert muss auf 1 gesetzt werden, um die Anzeige von .chm-Dateien im Intranet zu erlauben.

Die Anpassung betrifft alle Benutzerkonten des Computers.

Beim Starten des CONZEPT 16-Clients erscheint die Fehlermeldung: "Systemfehler: Font nicht gefunden".

Die Datei C16\_WIN.FON konnte nicht gefunden oder geladen werden. Folgende Gründe können dafür verantwortlich sein:

- **Die Datei befindet sich nicht im Arbeitsverzeichnis.**

Die Datei C16\_WIN.FON muss sich im Arbeitsverzeichnis des CONZEPT 16-Clients befinden. Häufig ist in der Verknüpfung zum Starten des Clients ein Arbeitsverzeichnis eingetragen, das vom Startverzeichnis abweicht.

- **Der Zeichensatz kann nicht geladen werden, weil bereits zu viele Zeichensätze installiert sind.**

Sind auf einem System zu viele Zeichensätze installiert (unter Windows 98 ca. 200), kann der CONZEPT 16-Zeichensatz nicht geladen werden. In diesem Fall müssen Zeichensätze, die nicht verwendet werden, gelöscht werden.

- **Der Zeichensatz kann nicht geladen werden, weil nur TrueType-Zeichensätze geladen werden.**

In dem Fenster der Schriftarten (Start / Einstellungen / Systemsteuerung / Schriftarten) ist in den Ordneroptionen (Menü: Extras / Ordneroptionen) auf der Notizbuchseite "TrueType-Schriftarten" der Eintrag "Nur TrueType-Schriftarten anzeigen" aktiviert. Von dem System werden dann nur TrueType-Schriften geladen. Diese Einstellung muss deaktiviert werden.

Warum erscheint beim Start von CONZEPT 16 die Meldung "Hauptfenster kann nicht eingerichtet werden"?

1. Die Datei C16\_TOOL.DLL befindet sich nicht im Verzeichnis von CONZEPT 16.
2. Die Datei C16\_TOOL.DLL hat nicht die korrekte Version.

Welche Umgebungsvariablen können beim Start von CONZEPT 16 angegeben werden?

Folgende Umgebungsvariablen können als Parameter über die Kommandozeile übergeben werden:

<u>/c16=&lt;pfad&gt;</u>	CONZEPT 16-Verzeichnis
<u>/c16cfg=&lt;pfad\dateiname&gt;</u>	Pfad für Konfigurationsdatei (*.cfg)
<u>/c16tmp=&lt;pfad&gt;</u>	Verzeichnis für temporäre Dateien
<u>/c16lang=DE EN TR</u>	Anzeigesprache im <u>Designer</u> , in CONZEPT 16-Meldungen, Menüaktionen und der textbasierten Oberfläche

/c16splashon=y|n Splash-Screen beim Start des Standard-Clients anzeigen  
Die Position dieser Angaben innerhalb der Kommandozeile ist ohne Bedeutung.

**Beispiel:**

```
c:\c16\c16_winc.exe * db_entw su /c16cfg=c:\c16\entw.cfg /c16tmp=c:\tmp  
/c16lang=EN
```

In diesem Beispiel wird die Datenbank db\_entw im Verzeichnis d:\ca1\entw mit dem Benutzer SU aufgerufen, wobei die Einstellungen aus c:\c16\entw.cfg benutzt und temporäre Dateien in c:\tmp gespeichert werden. Zusätzlich wird Englisch als Anzeigesprache verwendet.

Über die Funktion SysGetArg() können Kommandozeilenargumente abgefragt werden.

Wie kann die Umgebungsvariable C16TMP gesetzt werden?

1. Übergabe von /C16TMP= als Programmargument
2. Definition von C16TMP in der CONZEPT 16-Konfigurationsdatei \*.CFG
3. Definition von C16TMP als Umgebungsvariable (Environment)



Die Reihenfolge der Aufzählung entspricht der Priorität für den Fall, dass C16TMP mehrfach gesetzt wurde.

Welche temporären Dateien werden vom CONZEPT 16-Client angelegt?

Der Client legt temporäre Dateien im TEMP-Verzeichnis von Windows ab. Beim Schließen des Programms werden diese Dateien gelöscht. Wird ein Client nicht ordnungsgemäß geschlossen, können die Dateien nicht gelöscht werden. Beim Start eines Clients werden temporäre Dateien, die älter als ein Tag und nicht mehr im Zugriff sind, automatisch gelöscht.

Der Name der temporären Datei lautet immer c16\_<Nummer>.tmp. Wobei die Nummer achtstellig ist.

FAQ - Server

Häufige Fragen zum CONZEPT 16-Server

Die Fragen sind in folgende Bereiche aufgeteilt:

- Allgemeine Fragen
- Fragen zum Windows-Server
- Fragen zum Linux-Server
- Fragen zur Hot-Standby-Option

## FAQ - Server (Allgemeine Fragen)

### Allgemeine Fragen zum CONZEPT 16-Server

#### Welchen TCP/IP-Port verwendet der CONZEPT 16-Server?

Alle verwendeten Ports werden im FAQ zur Installation beschrieben. Die Ports, die vom CONZEPT 16-Server verwendet werden, sind ebenfalls im Abschnitt Architektur des Servers erläutert.

#### Welche Dateien werden vom CONZEPT 16-Server im Datenbankverzeichnis angelegt?

1. **<Datenbankname>.tl?** Im laufenden Betrieb werden insgesamt drei Transaktionslogdateien angelegt (Siehe Transaktionen während Sicherungsereignissen)
2. **<Datenbankname>.trs** Temporäre Transaktionsdatei
3. **<Datenbankname>.tmp** Temporäre Datei für die Prozedurverarbeitung durch den Datenbankserver
4. **<Datenbankname>.lgb** Log-Datei der Datenbank



Standardmäßig werden die Transaktionsdateien im Verzeichnis der Datenbank abgelegt. Bei den Serverparametern für die Datenbank kann aber ein alternativer Transaktionspfad angegeben werden. Ist dieser Pfad vorhanden, werden diese Dateien dort abgelegt.

Beim Start des CONZEPT 16-Servers erscheint "Initialisierungsfehler:

Verbindungsservice kann nicht eingerichtet werden".

Der CONZEPT 16-Server kann Ressourcen, die für die Kommunikation benötigt werden, nicht reservieren. In der Regel erscheint diese Meldung, wenn der CONZEPT 16-Server ein zweites Mal auf dem gleichen Computer gestartet wird. Es ist daher zu beachten, dass der CONZEPT 16-Datenbankserver nur einmal auf einem System gestartet werden kann.

#### Was besagt die Meldung "Lizenzdatei nicht gefunden" bzw. "Licensefile not found" beim Start des Servers?

Ab der Version 5.0 wird nur noch eine Lizenzdatei benötigt. Diese Datei mit dem Namen c16.lic muss im Daten-Verzeichnis des Servers vorhanden sein (siehe Speicherorte von Konfigurationsdateien). Die Clientprogramme benötigen keine Lizenzdatei mehr.

#### Wie kann man überprüfen, ob der USB-Dongle korrekt auf die USB-Schnittstelle gesteckt wurde?

Auf dem Lizenz-Dongle des CONZEPT 16-Servers befindet sich eine grüne LED. Diese leuchtet nur, wenn der Dongle korrekt aufgesteckt wurde und mit genügend Strom versorgt wird. Leuchtet diese LED nicht, muss die Schnittstelle, beziehungsweise die BIOS-Konfiguration des Mainboards überprüft werden.

#### Der CONZEPT 16-Server wird in der Serverauswahlliste nicht angezeigt.

- TCP/IP-Server werden nur dann in der Liste angezeigt, wenn ein entsprechender Eintrag in der Datei C16.HST vorhanden ist.
- Das Protokoll TCP/IP steht auf der Arbeitsstation nicht zur Verfügung.
- TCP/IP ist nicht korrekt installiert / konfiguriert.

## Kontakt

Im Dialog "Datenbank öffnen" steht statt der Größe der Datenbank "- unbekannt -". Der CONZEPT 16-Server kann die Datenbank nicht finden.

1. Die Datenbank befindet sich nicht auf dem angegebenen Laufwerk oder Verzeichnis.
2. Der Name der Datenbank ist nicht korrekt eingegeben.
3. Der Name der Datenbank wurde mit Erweiterung (.CA1) eingetragen.
4. Die Datenbank liegt auf einem lokalen Netzwerklaufwerk und der CONZEPT 16-Server wurde als Dienst unter Windows gestartet.
5. Der Pfadname wurde nicht gemäß den Konventionen des Serverbetriebssystems angegeben.

### *Beispiele:*

Server unter Windows: c:\c16\ca1\addresses

Server unter Linux: \home\usr\ca1\addresses oder /home/usr/ca1/addresses

Nach der Auswahl der Datenbank erscheint die Fehlermeldung "Datenbank ist gesperrt".

Auf der zu öffnenden Datenbank besteht eine Login-Sperre.

1. Im Bereich Datenbankservice wurde die Aktion Login sperren ausgeführt.
2. Die Datenbank wurde mit der A- Funktion LockDB() gesperrt.
3. Die primäre Datenbank wurde im Hot-Standby Betrieb nicht ordnungsgemäß geschlossen und der Datenbankserver erwartet die Freigabe für Rollback (siehe Servicefunktionen im Hot-Standby Betrieb).

Nach der Auswahl der Datenbank erscheint die Fehlermeldung "Keine Verbindung". Eventuell versucht sich der CONZEPT 16-Client mit einem CONZEPT 16-Server der Evaluierungslizenz zu verbinden. Falls dies der Fall ist, lautet die Lizenz des Servers "CD...EV". Es muss nun entweder ein Evaluierungs-Client verwendet oder der Datenbankserver mit Lizenz betrieben werden.

Nach der Auswahl der Datenbank erscheint die Fehlermeldung "Datenbank im Standby-Modus".

Die ausgewählte Datenbank ist für den Hot-Standby-Betrieb als Sekundärdatenbank eingerichtet und kann daher nicht geöffnet werden.

Nach der Auswahl der Datenbank erscheint die Fehlermeldung "Benutzerlimit erreicht".

1. Der Puffer der Datenbank ist zu klein. Pro Datenbankbenutzer sind mindestens 100 KB an Puffer zu reservieren (siehe auch: Speicherverbrauch des Servers).
2. Die maximale Benutzeranzahl des CONZEPT 16-Servers ist erreicht.

Beim Einfügen einer neuen Datenbank oder bei Änderungen einer bereits eingetragenen Datenbank erscheint die Meldung "Datenraumtabelle kann nicht geschrieben werden".

Der CONZEPT 16-Server kann in die Datei c16\_serv.ars (Datenraumtabelle) nicht schreiben. Diese Datei befindet sich im Verzeichnis des CONZEPT 16-Servers und beinhaltet unter anderem eine Aufstellung der beim Server eingetragenen

Datenbanken.

Wie groß muss der Datenbank-Puffer mindestens sein?

Siehe Speicherverbrauch des Servers.

Wie groß kann der Datenraum maximal werden?

Die Größe des Datenraums ist vom Betriebssystem abhängig, auf dem der CONZEPT 16-Server installiert ist. Je nach Betriebssystem ergeben sich folgende maximale Datenraumgrößen:

Serverplattform	Maximale Datenraumgröße	Maximale Datenbankgröße
Windows mit FAT 32	2 Gigabyte	16 Gigabyte
Windows mit NTFS	4 Terabyte	4 Terabyte
Linux Kernel 3.4	4 Terabyte	4 Terabyte

Nach dem Start des CONZEPT 16-Clients dauert es sehr lange, bis die Auswahlliste der verfügbaren CONZEPT 16-Server erscheint.

1. In der Hostdatei C16.HST ist eine große Anzahl von IP-Adressen oder Namen von Servern eingetragen.
2. Eines oder mehrere der in der C16.HST eingetragenen Systeme sind nicht im Netz erreichbar.

Bei der Auswahl einer Datenbank erscheint "Fehler bei Serverstart".

Der CONZEPT 16-Server kann möglicherweise den für die Datenbank eingetragenen Datenbankpuffer nicht anlegen. Der Datenbankpuffer ist größer als der frei verfügbare Hauptspeicher auf dem Datenbankserver. Der Puffer muss verkleinert oder mehr Hauptspeicher zur Verfügung gestellt werden.

Nach Auswahl der Datenbank erscheint "Transaktionsdatei kann nicht geöffnet werden".

Die Transaktionsdatei (<Datenbankname>.trs) kann nicht angelegt werden.

1. In dem Zielverzeichnis fehlt die Schreibberechtigung.
2. Das Zielverzeichnis existiert nicht.
3. Die Transaktionsdatei existiert bereits, ist aber noch von einem anderem Prozess im Zugriff.

Siehe auch: "Welche Dateien werden vom CONZEPT 16-Server angelegt?".

Kann eine Datenbank im laufenden Betrieb gesichert werden?

Ja, die Datenbank kann vom Datenbankserver für eine bestimmte Zeit für den Lesezugriff freigegeben werden. Die Benutzer können während des gesamten Zeitraums der Sicherung uneingeschränkt weiterarbeiten.

Für weitere Informationen siehe Sicherungsereignisse.

Was ist beim Betrieb des Servers auf einer virtuellen Maschine zu beachten?

Hinweise dazu befinden sich im Abschnitt Hinweise zum Betrieb auf virtuellen Maschinen.



FAQ - Server (Fragen zum Windows-Server)

Häufige Fragen zum Windows-Server

Kann beim Start des Clients der CONZEPT 16-Server automatisch gestartet werden?

In der Client-Konfigurationsdatei c16.cfg kann der automatische Start des lokalen Servers über den Parameter ServerAutostart realisiert werden.

Ab welcher Version kann CONZEPT 16 unter den verschiedenen Betriebssystemen eingesetzt werden?

<b>Betriebssystem</b>	<b>ab CONZEPT 16-Version</b>
Windows Server 2016	5.8.06
Windows 10	5.8.00
Windows Server 2012 R2	5.7.05
Windows 8.1	5.7.05
Windows Server 2012	5.7.00
Windows 8	5.7.00
Windows Server 2008 R2	5.5.00
Windows 7	5.5.00
Windows Server 2008	5.3.00
Windows Vista	5.2.00
Windows Server 2003	1.8.06
Windows XP	1.6.01

Wie kann das Beenden des CONZEPT 16-Server-Dienstes aus einem Skript heraus vorgenommen werden?

Unter Windows können Windows-Dienste mit dem Befehl net stop "<Name des Dienstes>" beendet werden. Für den CONZEPT 16-Server lautet die Anweisung demnach:

```
net stop "CONZEPT 16-Server"
```

Über die Anweisung net start werden alle unter Windows gestarteten Dienste aufgelistet.

Beim Start des CONZEPT 16-Servers als Dienst wird die Fehlermeldung "Der CONZEPT 16-Server hat den dienstspezifischen Fehler ... zurückgegeben" ausgegeben.

Die Nummer des "dienstspezifischen" Fehlers wird von Windows generiert und gibt keinen Aufschluss über das tatsächlich vorliegende Problem. Die Fehlermeldung des CONZEPT 16-Servers wird in den Log-Dateien des Servers festgehalten (siehe auch Architektur des Servers / Serverlogs). Diese befinden sich im Verzeichnis des CONZEPT 16-Servers.

Wie kann man den CONZEPT 16-Server ab Version 5.2 mit Einzelplatzlizenz über einen Client 5.0 oder früher erreichen?

Ab Version 5.2 kann im Einzelplatzbetrieb mit Clients der Versionen 5.0 und früher nur über die Loopback-Adresse erreicht werden (127.0.0.1 oder localhost). Ein \* kann nicht verwendet werden, da sich das \*-Protokoll bei der Version 5.2 zu den älteren Versionen geändert hat. Siehe auch Architektur des Servers und Lizenzmanagement.

### FAQ - Server (Fragen zum Linux-Server)

#### Häufige Fragen zum Linux-Server

Wie kann der CONZEPT 16-Server über ein Script gestartet werden?

Das folgende Skript kann zum Starten, Stoppen und zur Überprüfung des Status verwendet werden. Die Variable `c16_dir` muss entsprechend dem Installationsverzeichnis angepasst werden. Die Kommentarzeilen `# chkconfig ...` und `# decription ...` werden bei einer RedHat-Distribution zum automatischen Starten beim Booten der Maschine benötigt.

```
#!/bin/sh## Author: vectorsoft AG# chkconfig: 2345 55 25# description: start and stop CONZEPT 16
```

Wie kann der CONZEPT 16-Server unter einer SUSE-Distribution beim Booten des Systems automatisch gestartet werden?

Linux-Systeme starten in verschiedenen Run-Leveln. Runlevel 2 ist dabei der Multi-User Level. Ab diesem Runlevel sollte der CONZEPT 16-Server zur Verfügung stehen. Das Starten des Servers erfolgt dabei über ein Skript (siehe oben). Dieses Skript wird in das Verzeichnis `/etc/init.d` kopiert. Es kann jedes andere Skript verwendet werden, solange die Übergabeparameter `start` und `stop` korrekt verarbeitet werden. Anschließend kann das Skript mit der Anweisung `insserv` aktiviert werden.

```
/sbin/insserv -d $c16_script
```

In diesem Beispiel wird der CONZEPT 16-Server im voreingestellten Default-Runlevel des Systems gestartet. Dem Kommando `insserv` können andere Parameter übergeben werden, um der Server in einem bestimmten Runlevel zu starten (siehe die entsprechenden man-Pages).



Vor dem Start des Servers muss der Dongle-Treiber bzw. die Verbindung zum Internet vorhanden sein. Der Server muss nach den entsprechenden Daemon gestartet werden.

Wie kann unter Linux überprüft werden, ob auf einem System CONZEPT 16-Prozesse laufen?

Unter Linux kann mit dem `ps`-Kommando eine Liste der laufenden Prozesse erzeugt werden. Um festzustellen, ob ein CONZEPT 16-Prozess auf der Maschine gestartet wurde, wird auf der Konsole nachfolgendes Kommando angegeben.

```
ps -A | grep c16*
```

Ab welcher Kernel-Version des Linux-Systems kann der CONZEPT 16-Server eingesetzt werden?

Der CONZEPT 16-Server-Version ab der Version 5.7 setzt mindestens die Installation der Kernel-Version 2.6.25, 3.x oder 4.x voraus. Mit diesem Kernel wird auch der 64-Bit-Datenzugriff (Datenbanken größer 2 GB) unterstützt. Zudem wird die `glibc` in der Version 2.10 benötigt.



Ab Version 5.8.11 des CONZEPT 16-Servers wird die Kernel-Version 3.x oder 4.x vorausgesetzt. Zudem wird die `glibc` in der Version 2.17 benötigt.



Kernelversion 5.x wird ab Version 5.8.12 des CONZEPT 16-Servers unterstützt.



Unter Linux kann über das Kommando `uname -r` die Version des Kernels ermittelt werden. Die Version von der `glibc` kann mit dem Kommando `ldd --version`

ausgegeben werden.

Worauf ist beim Einsatz des CONZEPT 16-Servers unter Linux bezüglich der Prozessstruktur zu achten?

Der CONZEPT 16-Server startet für jede geöffnete Datenbank einen eigenen Prozess. Innerhalb der Prozesse wird für jeden Benutzer ein eigener Thread angelegt. Daher sind die prozessspezifischen Kernel-Limits insbesondere im Hinblick auf den Speicherverbrauch und die Anzahl der Filehandles wirksam.

Das Prozesslimit von 256 Filehandles reicht für maximal 200 Benutzer pro Datenbank, da jede TCP/IP-Verbindung einen Handle benötigt. Für eine höhere Benutzeranzahl muss dieses Limit entsprechend erhöht werden.



Mit dem Befehl `ulimit -a` werden alle gesetzten Limits angezeigt.

FAQ - Server (Fragen zur Hot-Standby Option)

Häufige Fragen zur Hot-Standby Option

Ab welchen CONZEPT 16-Versionen kann die Hot-Standby Option eingesetzt werden?

**Version**

Server	1.7.01
Client	3.6.10 und 4.4.12
Web-Schnittstelle	2.1.00
ODBC-Treiber	1.2.01
Windows Programmierschnittstelle	4.5.00

Gibt es Konstellationen, die zur Unterbrechung des Hot-Standby Betriebes führen?

Ja, 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 der Datenbank

Nach dem Einrichten von Hot-Standby passiert nichts. Ist etwas nicht korrekt konfiguriert?

Es muss kein Problem in der Konfiguration des Systems vorliegen. Das Hot-Standby System wird erst aktiv, wenn eine Datenbank von einem Benutzer geöffnet wird, für die die Hot-Standby Option über die Hot-Standby Einstellungen korrekt eingerichtet wurde.

Wird diese Datenbank zum ersten Mal geöffnet, findet zunächst eine Synchronisation der Datenbank vom Primärserver zum Sekundärserver statt, das heißt die Datenbank wird auf den Sekundärserver übertragen. Diese Synchronisation kann je nach Größe der Datenbank etwas Zeit beanspruchen. Der Erfolg der Synchronisation kann auf dem Sekundärsystem und in den Log-Dateien der Datenbanken überprüft werden. Die Hot-Standby Funktionalität kann jetzt auch über den Status der Datenbanken verifiziert werden. Dazu kann in dem Dialog "Datenbank öffnen" die Schaltfläche [Service] gedrückt werden. Im Datenbankstatus müssen dann zwei Zeilen angezeigt werden. Die Reihenfolge der Zeilen kann dabei unterschiedlich sein, je nachdem, ob der Status vom Primär- oder vom Sekundärserver aus ermittelt wird.

Master, geöffnetSlave, Standby

Kann die Datenbank zwischen dem Primär- und dem Sekundärserver kopiert werden?



Die Datenbanken dürfen **nicht** zwischen den Systemen kopiert werden! Die Übertragung der Datenbank zwischen den beiden Systemen erfolgt **immer** durch das Hot-Standby System!

Die Primär- und die Sekundärdatenbank sind mit einem Zeitstempel versehen. Immer wenn in die Primärdatenbank etwas geschrieben wird, wird ebenfalls der Zeitstempel aktualisiert. Beim Übertragen der Informationen in die Sekundärdatenbank erhält diese einen Zeitstempel der genau 100 Nanosekunden älter ist als der in der Primärdatenbank.

Da eine Abweichung von 100 Nanosekunden nicht durch eine Zeitmessung zustande

## Kontakt

kommen kann (die Granularität bei der Zeitmessung liegt bei ca. 1 Millisekunde), müssen Datenbanken mit einer Abweichung von 100 Nanosekunden synchron sein und die jüngere der beiden Datenbanken ist die Primärdatenbank.

Wird eine der Datenbanken auf das andere System kopiert, kommt es zu einer Situation, in der beide Zeitstempel gleich sind. Es kann nicht mehr unterschieden werden, welche die Primär- und welche die Sekundärdatenbank ist. Die Datenbank, die zuerst geöffnet wird erklärt sich zur Primärdatenbank. Anschließend wird die Datenbank auf dem anderen System gelöscht und die Synchronisation gestartet.

Wieso gibt es Einträge in der Log-Datei der Sekundärdatenbank, obwohl diese nur im Standby-Modus läuft?

Auch in der Log-Datei der Standby-Datenbank werden Eintragungen vorgenommen, wenn der Datenbankserver versucht den Status der Datenbank abzufragen oder den Hot-Standby Betrieb einleitet. Wendet sich beispielsweise ein Benutzer an den Primärserver entstehen folgende Einträge:

### *Primärdatenbank*

```
HSB Master 2008-07-25 07:56:18 Database opened (...) [...] HSB Master 2008-07-25 07:56:19
```

### *Sekundärdatenbank*

```
HSB Slave 2008-07-25 07:56:18 Database opened (...) [...] HSB Slave 2008-07-25 07:56:19 C
```

Versucht sich allerdings ein Benutzer bei der Sekundärdatenbank anzumelden, entstehen folgende Einträge.

### *Primärdatenbank*

```
HSB Slave 2008-07-25 07:58:06 Connect from 10.1.0.8 HSB Slave 2008-07-25 07:58:06 Disconne
```

### *Sekundärdatenbank*

```
HSB Master 2008-07-25 07:58:06 Connect to 10.1.0.7 HSB Master 2008-07-25 07:58:06 Discor
```

Der Server mit der Sekundärdatenbank überprüft, ob der Server mit der Primärdatenbank noch läuft. Dazu baut er eine Verbindung zum Primärserver auf ("Connect to"). Da der Primärserver noch existiert, wird anschließend die Verbindung wieder getrennt und der Benutzer abgewiesen ("Disconnect from").

## FAQ - Web-Schnittstelle

### Häufige Fragen zur Web-Schnittstelle

Kann die Datei c16\_web.dll umbenannt werden?

Ja. Der Name der Datei kann beliebig umbenannt werden, lediglich die Dateierweiterung (.DLL) muss erhalten bleiben.

Alle Dateien, die von der Web-Schnittstelle angelegt bzw. gelesen werden, erhalten dann das gleiche Präfix. Wird die Datei zum Beispiel in SHOP.DLL umbenannt, werden die Protokolldateien entsprechend mit SHOP.<Datum>.LOG angelegt. Ebenso werden alle anderen Dateien (CFG, INI usw.) mit SHOP anstelle von c16\_web erwartet.

Unterstützt die Web-Schnittstelle die Hot-Standby-Option des CONZEPT 16-Servers?  
Ab der Version 2.1.00 unterstützt die Web-Schnittstelle den Betrieb mit Hot-Standby.

Dazu werden in der cfg-Datei bei dem Eintrag c16\_server beide Server angegeben.

### Beispiel

c16\_server = TCP:10.1.3.1+TCP:10.1.3.2

Die maximale Wartezeit beim Verbinden mit der Datenbank wird im allgemeinen Teil der cfg-Datei (also global) eingestellt. Der Wert kann zwischen einer und 60 Sekunden betragen.

### Beispiel

c16\_connect\_timeout\_s = 2

Warum werden die Befehle in den HTML-Seiten nicht ausgeführt?

- Die Befehle innerhalb der HTML-Seiten müssen in Großbuchstaben geschrieben werden, damit sie von der Schnittstelle erkannt werden können. C16.Call wird nicht erkannt, C16.CALL wird erkannt.
- Innerhalb der Web-Schnittstelle können nur A+ Prozeduren aufgerufen werden. Wird mit einem C16.CALL-Aufruf eine A- Prozedur aufgerufen, wird dieser Aufruf ignoriert. Solche Prozeduren können auch nicht mit der Anweisung CallOld() innerhalb von A+ Prozeduren aufgerufen werden. Solche Aufrufe werden ebenfalls ignoriert.

Beim Aufruf eines Links innerhalb der Web-Anbindung wird immer die Startseite zurückgegeben.

- Das HTTP ist ein zustandsloses Protokoll, d. h. mit Hilfe dieses Protokolls kann kein Status eines Datenbankbenutzers gespeichert werden. Statt dessen wird von der Web-Schnittstelle ein Benutzer angelegt und die notwendigen Informationen bei diesem Benutzer abgelegt. Der Benutzer wird auf Grund einer Nummer identifiziert, die in jedem Link enthalten sein muss. Ist diese Nummer nicht vorhanden, wird angenommen, dass ein neuer Benutzer die Schnittstelle aufgerufen hat, dieser bekommt natürlich die Startseite übermittelt.

Die Links innerhalb der Applikation müssen die Benutzer-ID beinhalten. Die Benutzer-ID wird innerhalb der HTML-Seite mit der Anweisung C16.UID() zurückgegeben. Links, die mit der Anweisung C16.URL() erzeugt werden, verfügen bereits über eine Benutzer-ID.

- Die Benutzer-ID kann entweder im Pfad des Links oder als Parameter des Links übergeben werden. Ist die IIS Version 3.0 im Einsatz, muss die Benutzer-ID im Parameter angegeben werden. Dies kann in der c16\_web.cfg mit dem Eintrag web\_uid\_mode = query erfolgen.

Es wird eine Fehlerseite mit der Meldung "CONZEPT 16 error" zurückgegeben. Je nach HTTP-Status und Fehlerwert hat die Meldung folgende Bedeutung:

### • HTTP Status 503 und Fehlerwert 50026

1. Die maximale Benutzerzahl des CONZEPT 16-Servers ist erreicht.
2. Das in der Konfigurationsdatei angegebene Verbindungslimit (web\_max\_sessions) wurde überschritten.

In der Protokoll-Datei der Applikation erfolgt ein entsprechender Eintrag: Session refused - connection limit from <IP-Adresse> reached. Das aktuelle Limit wird in Klammern angegeben.

### • HTTP Status 503 und Fehlerwert 50025

Die in der Konfigurationsdatei angegebene maximale Anzahl der Web-Benutzer (web\_max\_connections) wurde überschritten. In der Log-Datei der Applikation erfolgt ein entsprechender Eintrag: Session refused - connection limit reached. Das aktuelle Limit wird in Klammern angegeben.

### • HTTP Status 503 und Fehlerwert 50032

Die Web-Schnittstelle kann zum CONZEPT 16-Server oder der Datenbank keine Verbindung aufbauen. Die möglichen Ursachen können der Protokoll-Datei der Applikation entnommen werden:

#### **Database connect failed -801**

Zum CONZEPT 16-Server kann keine Verbindung hergestellt werden.

#### **Database connect failed -803**

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

#### **Database connect failed -811**

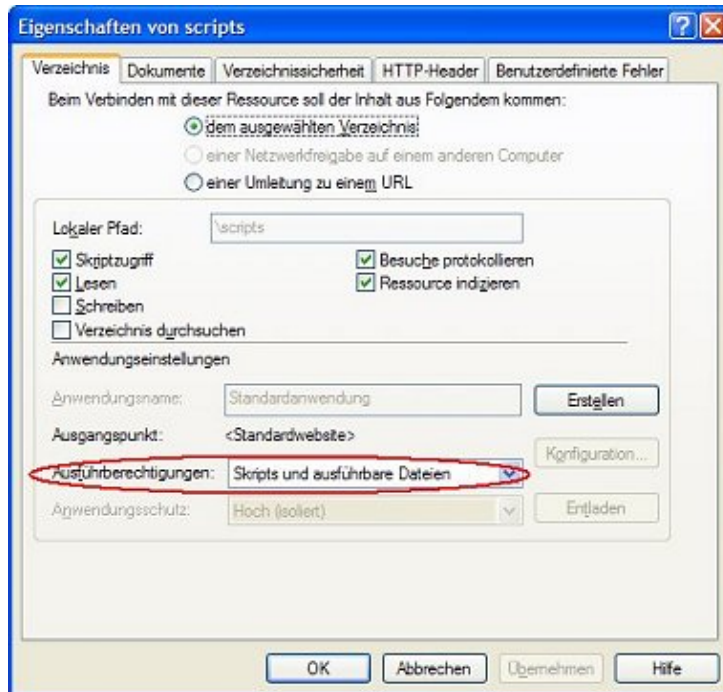
Der angegebene Benutzer ist nicht vorhanden, das Kennwort ist nicht korrekt oder dem Datenbankbenutzer fehlen die Rechte von außerhalb auf die Datenbank zuzugreifen (Benutzer Programmberechtigungen - Externer Zugriff).

Die maximale Anzahl der in der Konfigurationsdatei angegebenen Verbindungen zur Datenbank (c16\_max\_connections) wurde überschritten. In diesem Fall erfolgt kein Eintrag in der Protokoll-Datei.

Warum wird beim Zugriff auf die Web-Schnittstelle immer die c16\_web.dll heruntergeladen?

## Kontakt

Bei der Konfiguration des IIS muss in den Eigenschaften des Scripts-Verzeichnisses als Ausführungsberechtigung "Skripts und ausführbare Dateien" angegeben werden. Ist das nicht der Fall, wird die Datei c16\_web.dll heruntergeladen.





## Kontakt

FAQ - Externe Windows Programmierschnittstelle

Häufige Fragen zur externen Windows Programmierschnittstelle

Unterstützt die Programmierschnittstelle die Hot-Standby-Option des CONZEPT 16-Servers?

Ab der Version 4.5.00 der Programmierschnittstelle wird der Betrieb mit Hot-Standby unterstützt. Analog zum Client werden bei der Funktion C16\_OpenArea() in aServerName beide Server angegeben (beispielsweise "10.1.0.1+10.1.0.2"). In aAreaName wird dann der symbolische Datenbankname übergeben. Der Verbindungs-Timeout ist dabei fest auf 3 Sekunden eingestellt. Als neues Resultat von C16\_OpenArea() wird C16ERR\_AREA\_STANDBY zurückgeliefert, wenn die Datenbank des Zielservers im Standby-Modus ist.

## FAQ - ODBC-Schnittstelle

### Häufige Fragen zur ODBC-Schnittstelle

Wie kann die Version des installierten CONZEPT 16-ODBC-Treibers ermittelt werden?  
Auf der Notizbuchseite "Treiber" des ODBC-Datenquellen-Administrators von Windows existiert in der Liste der installierten ODBC-Treiber der Eintrag vectorsoft CONZEPT 16-Treiber, in der Spalte "Version" wird die Release des Treibers angezeigt.

Wie kann erreicht werden, dass nur bestimmte Benutzer über die ODBC-Schnittstelle auf die Datenbank zugreifen können?

In den Programmberechtigungen des Datenbankbenutzers kann der Zugriff über die Option "Externer Zugriff" gesteuert werden. Ist die Option nicht gesetzt, kann der Benutzer nicht über die ODBC-Schnittstelle auf die Datenbank zugreifen.



Ist die Option nicht gesetzt, ist der Zugriff auf die Datenbank auch über die Web-Schnittstelle, Programmierschnittstelle, SOA-Service und über die Funktion DbConnect() nicht möglich.

Wie können Abfragen optimiert werden?

Bei Abfragen über das SELECT-Statement werden, falls möglich, Schlüssel zur Optimierung des Zugriffs verwendet. Schlüssel über alpha-Felder können nur verwendet werden, wenn keine Attribute (Definitionsgröße, Groß-/Kleinwandlung usw.) dem Schlüsselfeld zugeordnet sind. In der Regel sollte ein Schlüssel definiert werden, wenn eine Bereichsabfrage oder eine Abfrage auf einen JOIN verwendet wird, deren Bedingung ein alpha-Feld enthält.

Unterstützt die ODBC-Schnittstelle die Hot-Standby-Option des CONZEPT 16-Servers?  
Ab der ODBC-Treiber-Version 1.2.01 wird der Betrieb mit Hot-Standby unterstützt. Analog zum Client werden bei der Datenquelle beide Server angegeben (beispielsweise 10.1.0.1+10.1.0.2). Der Verbindungs-Timeout ist dabei fest auf 3 Sekunden eingestellt.

Beim Zugriff auf eine CONZEPT 16-Datenbank über den ODBC-Treibermanager werden nicht alle Felder einer Datei angezeigt.

Die meisten Abfrage-Werkzeuge wie zum Beispiel MS-Query können nicht mehr als 256 Spalten pro Tabelle verwalten. Einige wenige Programme reagieren sogar mit einem Absturz, wenn in einer Tabelle mehr als 256 Spalten existieren.

CONZEPT 16 kann bis zu 5.000 Spalten zurückgeben. Voraussetzung dafür ist, dass das Abfrage-Werkzeug mehr Spalten verarbeiten kann und der Eintrag MaxColSupport für die entsprechende Datenquelle in der Windows Registrierungsdatenbank auf den Wert der maximal verarbeiteten Spalten gesetzt wird.

Nach der Auswahl der Datenquelle und Eingabe des Datenbankbenutzers erscheint die Meldung "Connect to server failed".

1. Der CONZEPT 16-Server, auf dem die ODBC-Schnittstelle reserviert ist, ist nicht erreichbar (Nicht gestartet, falsche IP-Adresse).
2. Die Angaben des CONZEPT 16-Servers (Protokoll,Server) in der ODBC-Datenquelle sind nicht korrekt.
3. Es wird ein CONZEPT 16-Server eingesetzt, dessen Release kleiner 1.5.04 ist.

## Kontakt

Nach der Auswahl der Datenquelle und Eingabe des Datenbankbenutzers erscheint die Meldung "Database not found".

Der CONZEPT 16-Server konnte die Datenbank nicht öffnen. Evtl. ist die Datenbank nicht vorhanden oder nicht korrekt beim Treibermanager eingetragen.

Nach der Auswahl der Datenquelle und Eingabe des Datenbankbenutzers erscheint die Meldung "Invalid user account or password".

1. Der angegebene Datenbankbenutzer existiert nicht.
2. Das Kennwort ist nicht korrekt
3. Dem Datenbankbenutzer fehlen die Rechte über ODBC die Datenbank zu öffnen. In den Programmberechtigungen des Datenbankbenutzers kann der Zugriff über die Option "Externer Zugriff" gesteuert werden. Ist die Option nicht gesetzt, kann der Benutzer nicht über die ODBC-Schnittstelle auf die Datenbank zugreifen.

Nach Auswahl der Datenquelle und Angabe des Datenbankbenutzers erscheint die Meldung "User limit is reached".

Der Benutzer wurde bei der Anmeldung abgewiesen, da die maximale Benutzeranzahl des CONZEPT 16-Servers erreicht ist.

Nach der Anmeldung an die Datenbank erfolgt die Meldung "Database is locked". Die Meldung kann unterschiedliche Ursachen haben:

- **Die Datenbank ist gesperrt.**

Die Datenbank ist aufgrund einer Diagnose mit Recover oder durch den 4.0 kompatiblen Befehl LockDb() gesperrt.

- **Die Datenbank befindet sich im Standby-Modus.**

Die Datenbank kann nicht geöffnet werden, da diese vom CONZEPT 16-Server im Standby-Modus geöffnet ist. (siehe Hot-Standby).

- **Die Datenbank hat die falsche Version.**

Es wird versucht mit der aktuellen Version des ODBC-Treibers auf eine Datenbank der Version 4.6 oder früher zuzugreifen. Aufgrund der Änderungen bei den Feldtypen kann der ODBC-Treiber nur mit Datenbanken der Version 4.7 und größer kommunizieren.

Beim Speichern oder Löschen eines Datensatzes erscheint die Meldung "Record lock violation".

Der Datensatz ist bereits von einem anderen Benutzer gesperrt.

Beim Einfügen eines Datensatzes erscheint die Meldung "Unique key violation".

Der Datensatz konnte nicht eingefügt werden, da bereits ein Datensatz mit einem identischen eindeutigen Schlüsselwert existiert.

Bei einer Abfrage oder Schreiben eines Datensatzes erscheint die Meldung "Numeric value out of range".

Es wurde ein Wert übergeben, der sich außerhalb des gültigen Wertebereiches eines CONZEPT 16-Datenbankfeldes befindet. In diesem Fall wird kein Datensatz in der

Datenbank geschrieben.

Beim Einfügen / Ändern eines Datensatzes erscheint die Meldung "Data truncated column name".

Einem Datenbankfeld vom Typ alphanumerisch wurde eine zu lange Zeichenkette zugewiesen. Die Zeichenkette wird abgeschnitten und der Datensatz gespeichert. Das betreffende Datenbankfeld wird bei der Fehlermeldung hinter dem Doppelpunkt ausgegeben:

Data truncated column: ADR\_aFirm

Die Meldung besagt, dass die Spalte 'ADR\_aFirm' nicht die komplette Zeichenkette aufnehmen kann.

Bei einer Abfrage oder Datensatzoperation erscheint die Meldung "No such table or object".

Es wurde eine Tabelle angegeben, die in der Datenbank nicht existiert. Es ist zu beachten, dass Namen in CONZEPT 16 Sonderzeichen beinhalten können. Diese Sonderzeichen werden unter ODBC möglicherweise anders dargestellt. Ein "." im Namen unter CONZEPT 16 wird in einen "\_" in ODBC umgewandelt.

Bei einer Abfrage oder Datensatzoperation erscheint die Meldung "Invalid column name".

Es wurde eine Spalte angegeben, die in der Datenbank nicht existiert. Der Name der Spalte, welche nicht referenziert werden konnte, wird bei der Fehlermeldung hinter dem Doppelpunkt ausgegeben:

Invalid column name: 'ADR\_aFirm'

Die Meldung besagt, dass die Spalte 'ADR\_aFirm' nicht gefunden wurde.

Bei Verarbeitung der SQL-Befehle "Insert", "Update" und "Delete" erscheint die Fehlermeldung "Insufficient user rights".

Dem Datenbankbenutzer fehlen die entsprechenden Rechte, um die Anweisungen auszuführen. Die Dateiberechtigungen welche dem Benutzer hinterlegt sind (**Benutzerpflege / Dateirechte**) werden bei dem Zugriff via ODBC ausgewertet.

Die ODBC-Schnittstelle des CONZEPT 16-Clients lässt sich nicht initialisieren (Der Befehl OdbcInit gibt als Ergebnis false zurück).

Es kann nur mit einer Enterprise- oder Developer-Edition auf eine ODBC-Datenquelle zugegriffen werden.

Wie ist eine Wertzuweisung für die CONZEPT 16-Feldtypen vorzunehmen?

- **Logisch**

- 0 für false, 1 für true

- **Gleitkomma, Dezimal**

- 10.99 für 10,99

- **Zeit**

'10:02' für 10 Uhr 02

Verwendet wird das Format '*hh:mm[:ss]*'

• **Datum**

'2005-02-01' für 01.02.2005

Verwendet wird das Format '*jjjj-mm-dd*'

• **Alphanumerisch**

'Dies ist ein String'

• **Ganzzahlig**

200

**Beispiel:**

```
UPDATE ADR_F_CUSTOMERS SET ADR_dDate = '2005-02-01' WHERE ADR_aFirmDescr = 'vectorsoft'
```

Welche SQL-Datentypen werden unterstützt?

Folgende Datentypen werden unterstützt:

<b>CONZEPT</b>	<b>SQL-Datentyp</b>
<b>16-Datentyp</b>	
Alphanumerisch	CHARACTER VARYING(n) oder VARCHAR(n) n maximal 4096 Zeichen
Ganzzahlig 64	NUMERIC
Ganzzahlig lang	LONG
Ganzzahlig kurz	SMALLINT
Dezimal	NUMERIC maximale Genauigkeit von 58 signifikanten Stellen.
Gleitkomma	DOUBLE
Logisch	BIT
Zeit	TIME
Datum	DATE

## FAQ - Internetbasierte Lizenzen

### Häufige Fragen zu internetbasierten Lizenzen

Wo **müssen** internetbasierte Lizenzen eingesetzt werden?

Lizenzen mit Softwareschutz (internetbasierte Lizenzen) müssen überall dort verwendet werden, wo keine Schnittstelle für ein Dongle vorhanden ist oder bei virtuellen Servern.

Welche Voraussetzungen müssen gegeben sein, damit eine internetbasierte Lizenz verwendet werden kann?

Die Voraussetzungen sind im Abschnitt Lizenz mit Softwareschutz beschrieben.

Wie nimmt der CONZEPT 16-Server Kontakt mit den Lizenzservern auf?

Die Verbindung zu einem der Lizenzserver erfolgt über das Internet. Abhängig davon, wie die Verbindung zum Internet hergestellt wird, müssen verschiedene Bedingungen beachtet werden:

- **Kein Proxy**

Die Verbindung wird entweder durch die Einwahl des Rechners beim Internet Service Provider oder über einen Standard Gateway hergestellt. In beiden Fällen muss lediglich gewährleistet sein, dass die lokale Firewall die Kommunikation zulässt.

- **HTTP-Proxy**

Erfolgt die Verbindung über einen HTTP-Proxy, muss der Name bzw. die IP-Adresse und der Port (Standard ist der Port 8080) des Proxys auf dem HTTP-Anfragen von dem Proxy entgegengenommen werden, in den Einstellungen angegeben werden. Verlangt der Proxy eine Authentifizierung, muss ein Benutzer und ein Kennwort angegeben werden. Als Authentifizierungsmethode wird nur Basic unterstützt.

- **SOCKS-Proxy (Version 4a)**

Bei der Verwendung eines SOCKS-Proxy der Version 4a muss ebenfalls der Name bzw. die IP-Adresse und der Port (Standard ist der Port 1080) angegeben werden. Ist eine Authentifizierung erforderlich, erfolgt diese nur über den Benutzer. Dieser wird als User-Id an den Proxy gesendet.

- **SOCKS-Proxy (Version 5)**

Wird ein SOCKS-Proxy der Version 5 verwendet, muss ebenfalls der Name bzw. die IP-Adresse und der Port (Standard ist der Port 1080) angegeben werden. Ist eine Authentifizierung erforderlich, erfolgt diese mit dem angegebenen Benutzer und Passwort. Die Authentifizierung erfolgt über das Username/Password-Verfahren.

Wie kann eine Regel in einem Proxy / einer Firewall definiert werden, damit der CONZEPT 16-Server mit den vectorsoft Lizenzservern kommunizieren kann?

Die Einstellungen sind je nach verwendetem Proxy oder der verwendeten Firewall sehr unterschiedlich, deshalb können hier nur Hinweise zu verschiedenen Einstellungen gegeben werden.

## Kontakt

Der CONZEPT 16-Server kommuniziert über das HTTP-Protokoll mit den Lizenzservern. Es muss somit lediglich eine Regel für dieses Protokoll erstellt werden.

Die Kommunikation geht immer vom CONZEPT 16-Server aus. Die Quelle der Kommunikation kann somit der Rechner auf dem der CONZEPT 16-Server gestartet ist und der Prozess "CONZEPT 16-Server" eindeutig definiert werden. Ziel der Kommunikation ist immer einer der bislang vier vectorsoft Lizenzserver. Diese können namentlich (sina1.c16.net/\*, sina2.c16.net/\*, sina3.c16.net/\*, sina4.c16.net/\*) angegeben werden.

Zusätzliche Einschränkungen zum Beispiel über den Inhalt der Kommunikation sind nicht sinnvoll, da diese verschlüsselt erfolgt. Eine Einschränkung auf das verwendete Programm (Agent) oder weitere Kopf-Einträge sollte vermieden werden, da eine eindeutige Zuordnung bereits durch Rechner und Prozess der Kommunikationsquelle erfolgt ist.

In der Logdatei steht "Successful connect to License Server ...". Es wird aber keine internetbasierte Lizenz verwendet?

Die Lizenzserver dienen nicht ausschließlich zur Validierung von internet basierten Lizenzen. Über sie kann auch die Aktualität der verwendeten Lizenzdatei (c16.lic) überprüft werden. Dabei wird die Lizenznummer an einen der Lizenzserver übertragen. Gibt es eine aktuellere Lizenzdatei, wird diese an den CONZEPT 16-Server übermittelt, der dann die vorhandene ersetzt. Dies erspart im Falle einer Benutzererweiterung oder einer anderen Änderung an der Lizenz, die Datei manuell zu ersetzen. Dies ist im besonderen dann wichtig, wenn eine zeitlich beschränkte Lizenz eingesetzt wird.

Es wird dazu nur die Lizenznummer übertragen. Die Kommunikation zum Abgleich der Lizenzdatei kann unterbunden werden. Die betreffende Einstellung befindet sich in der Datenraumtabelle.

Der CONZEPT 16-Server startet mit der Fehlermeldung "Connect to any License Server failed (...)".

Es konnte keiner der vectorsoft Lizenzserver erreicht werden. Möglicherweise ist die Kommunikation nicht möglich, da eine Firewall oder ein Proxy die Kommunikation verhindert oder bereits die Verbindung zum Internet scheitert (Internet Service Provider nicht erreichbar, Verbindung zum Standard Gateway nicht möglich usw.). Sind andere Verbindungen ins Internet möglich (zum Beispiel der Aufruf von Web-Seiten), können die Proxy-Einstellungen (siehe "Wie nimmt der CONZEPT 16-Server Kontakt mit den Lizenzservern auf?") überprüft werden.

### FAQ - Installation

#### Häufige Fragen zur Installation

Wann wird eine neue Lizenzdatei (c16.lic) benötigt?

Eine neue Lizenzdatei wird benötigt, wenn Änderungen an der Lizenz durchgeführt wurden. Dies ist zum Beispiel der Fall, wenn die Anzahl der Benutzer erhöht oder nachträglich eine Lizenz-Option erworben wurde. Die neue Lizenzdatei wird ebenfalls benötigt, wenn ein Versionswechsel in der zweiten Nummer des CONZEPT 16-Programmstandes (zum Beispiel der Wechsel von Version 5.2.03 auf Version 5.3.00) erfolgt.

Die Änderung kann im laufenden Betrieb erfolgen. Der CONZEPT 16-Server muss nicht neu gestartet werden. Die alte Lizenzdatei wird einfach durch die neue Datei ersetzt (siehe Speicherorte von Konfigurationsdateien). Innerhalb weniger Minuten stehen dann die Änderungen zur Verfügung.

Welche TCP/IP-Ports benutzt CONZEPT 16 zur Kommunikation?

Folgende Ports werden verwendet:

- 4721** Der Debugger stellt den Port 4721 zur Kommunikation zur Verfügung. Der Port muss nur dann freigegeben werden, wenn der CONZEPT 16-Client und der Debugger auf unterschiedlichen Maschinen gestartet wurden.
- 4722** Auf dem Port 4722 nimmt der Manager-Prozess die Datenbank Anfragen 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.
- 4729** Über den Port 4729 wird der CONZEPT 16-Printprocessor von den Druckertreibern angesprochen. Eine Freigabe des Ports muss nur dann erfolgen, wenn der Printprocessor auf einer anderen Maschine installiert wird, als die Druckertreiber.
- 4741** 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. Daher 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 Standardport zur Administration des Servers über einen Web-Browser. Der Port kann in der Einstellung WebSvcPort 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.
- 1583** Dieser Port wird zur Kommunikation zwischen dem ODBC-Client und dem ODBC-Treiber verwendet.

Bei einer bestehenden TCP/IP-Verbindung zu dem CONZEPT 16-Server kann durch die Eingabe des Kommandozeilen-Befehls `netstat -p tcp` ebenfalls der verwendete Port



ermittelt werden.

Wie wird die Hot-Standby-Option des CONZEPT 16-Servers installiert?  
Siehe Installation der Hot-Standby Option.

Der CONZEPT 16-Server kann nicht als Dienst eingerichtet werden.  
Wird der CONZEPT 16-Server auf Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, Windows Server 2012 R2, Windows 10 oder Windows Server 2016 installiert, kann es zu Fehlern während der Installation oder beim Starten des CONZEPT 16-Servers kommen, wenn der Benutzer sich nicht mit Administratorrechten an das System angemeldet hat.

Dem Benutzer kann beispielsweise das Schreiben in das Root-Verzeichnis des Festplattenlaufwerks oder das Einrichten von Diensten untersagt sein.

Die Reinstallation von CONZEPT 16-Lizenzen muss ebenfalls mit Administratorrechten erfolgen.

Der USB-Dongle wird vom Betriebssystem nicht automatisch erkannt.  
Der USB-Dongle verfügt über eine LED, die leuchtet, sobald der Dongle mit Strom von der Schnittstelle versorgt wird.

Beginnt die LED nicht zu leuchten, ist möglicherweise die USB-Schnittstelle im BIOS des Computers deaktiviert, nicht angeschlossen oder es gibt einen Hardwarekonflikt.

Bei der Installation konnte der Druckprozessor nicht gestartet werden.  
Der Druckprozessor benötigt eine Konfigurationsdatei, die bei der Installation nicht automatisch angelegt wird. Die Datei c16\_ppcsvc.cfg kann aus der Hilfe kopiert und angepasst werden.

Die genaue Fehlermeldung kann der Protokolldatei des Druckprozessors entnommen werden.

## FAQ - Listen

Häufige Fragen zu Listen in CONZEPT 16

Die Fragen sind in folgende Bereiche aufgeteilt:

- Fragen zur DataList
- Fragen zur RecList

## FAQ - DataList

### Fragen zur DataList

Wie wird das DataList-Objekt mit Daten gefüllt?

Zum Füllen eines DataList-Objekts mit Inhalten stehen die Befehle WinLstDatLineAdd() und WinLstCellSet() zur Verfügung.

Mit dem Befehl WinLstDatLineAdd() wird eine neue Zeile im DataList-Objekt erzeugt. Die erste Spalte kann bereits mit diesem Befehl mit Inhalt gefüllt werden. Um weitere Spalten einer Zeile mit Inhalt zu füllen wird der Befehl WinLstCellSet() verwendet.

### Beispiel:

```
// Liste mit fünf Zeilen füllen for tCnt # 1;loop Inc(tCnt);while (tCnt <= 5){ // Zeile erzeugen
```

In diesem Beispiel besteht die Liste aus zwei Spalten in denen ganze Zahlen angegeben werden können. Der Typ der Spalte wird beim Entwurf der Liste im DataList-Editor angegeben.

Können Daten in einem DataList-Objekt direkt geändert werden?

Mit der Funktion WinLstEdit() ist das Ändern von Feldern innerhalb eines DataList-Objekts möglich.

Eine Spalte im DataList-Objekt soll in Abhängigkeit vom Wert eines Feldes eine andere Farbe erhalten.

Um dies zu realisieren muss in dem Ereignis EvtLstDataInit des DataList-Objekts eine Funktion definiert werden, in der die Überprüfung des Wertes und das Setzen der Spaltenfarbe vorgenommen wird. Im nachfolgendem Beispiel wird die Firmenbezeichnung mit der Farbe rot dargestellt, wenn der Wert in der zweiten Spalte auf true steht.

### Beispiel:

```
sub EvtLstDataInit ( aEvt : event; // Ereignis aID : bigint; // Datensatz-ID oder Zeilennummer
```

Wie kann ermittelt werden, welcher Datensatz im DataList-Objekt selektiert ist?

Über die Eigenschaft CurrentInt des DataList-Objekts kann zu jedem Zeitpunkt die Nummer der selektierten Zeile ermittelt werden.

### Beispiel:

```
tCurrentLine # $DataList->wpCurrentInt;
```

Wird in dieser Eigenschaft der Wert 0 zurückgegeben ist keine Zeile selektiert.

Zum Auslesen des Zeileninhalts der selektierten Zeile muss nicht die Zeilennummer ermittelt werden. Wird beim Befehl WinLstCellGet() die Option WinLstDatLineCurrent angegeben, wird die selektierte Zeile gelesen.

Wie kann eine bestimmte Zeile im DataList-Objekt selektiert werden?

Über die Eigenschaft CurrentInt des DataList-Objekts kann die Nummer der zu selektierenden Zeile angegeben werden.

### Beispiel:

```
// erste Zeile selektieren$DataList->wpCurrentInt # 1;
```

Wie kann die Sortierung eines DataList-Objekts geändert werden?

Die Informationen innerhalb eines DataList-Objekts können nach jeder Spalte sortiert werden. Die Sortierung muss in der Eigenschaft ClmSortFlags der entsprechenden Spalten aktiviert werden. In der Beispiel-Datenbank befindet sich dazu das Beispiel CteDataList.

Im DataList-Objekt werden vertikale und horizontale Linien gezeichnet. Wie können diese Linien ausgeblendet werden?

Um die Linien auszublenden muss der Eigenschaft ColGrid der Wert \_WinColUndefined zugewiesen werden.

Der Anwender ändert in der Applikation die Reihenfolge der Spalten. Wie kann erreicht werden, dass diese Änderung bei Verlassen des Dialoges bzw. der Applikation erhalten bleibt?

Die Position einer Spalte kann über die Eigenschaft ClmOrder ermittelt und gesetzt werden. Um die Aufgabenstellung zu lösen, wird in der Ereignisfunktion EvtClose die Position aller Spalten des DataList-Objekts ermittelt und in eine Datei geschrieben.

### Beispiel:

```
sub EvtClose( aEvt : event; // Ereignis) : logic;local{ tObj : int;}{ // Abarbeiten der Spalte
```

Damit über die Funktion WinInfo() die Spalten in der Reihenfolge ihrer Anzeige ermittelt werden können, muss die Eigenschaft OrderPass des DataList-Objekts auf den Wert \_WinOrderShow gesetzt werden.

Beim Laden des Fensters wird in der Ereignisfunktion EvtInit die Datei gelesen und die Eigenschaft ClmOrder gesetzt.

Wie kann verhindert werden, dass der Anwender die Reihenfolge der Spalten verändert?

Wird der Eigenschaft LstStyle der Wert \_WinLstHeader zugewiesen, ist das Verändern der Reihenfolge nicht mehr möglich. Eine Änderung der Spaltenbreite kann noch vorgenommen werden.

Können in einem DataList-Objekt Bilder angezeigt werden?

In einem DataList-Objekt können keine Bilder angezeigt werden.

Die Ausnahme bilden Icons. Werden in einer Spalte der Liste ganzzahlige Werte (Ganzzahlig kurz, Ganzzahlig lang oder Ganzzahlig 64) angezeigt, kann statt der Zahl ein Icon aus einem Symbolsatz (Tile) angezeigt werden. Die Zahl entspricht der Position des Symbols innerhalb des Tile.

Wenn ein Symbol angezeigt werden soll, muss die Eigenschaft ClmTypeImage auf \_WinClmTypeImageC16 oder \_WinClmTypeImageUser gesetzt werden. Bei \_WinClmTypeImageC16 wird das CONZEPT 16 Tile verwendet. Im anderen Fall wird das benutzerdefinierter Tile verwendet, das beim DataList-Objekt in der Eigenschaft

TileNameUser angegeben wurde.

## FAQ - RecList

### Fragen zum RecList-Objekt

Nach einem Update auf das RecList-Objekt verändert sich die Position des selektierten Datensatzes. Wie kann das verhindert werden?

Soll der selektierte Datensatz nach einem Update die Position in dem RecList-Objekt beibehalten, muss die Funktion WinUpdate() wie folgt verwendet werden:

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

In welcher Reihenfolge werden die Ereignisse eines RecList-Objekts verarbeitet?

Siehe Ereignisse eines RecList-Objekts.

Wie kann über eine Prozedur in einem RecList-Objekt zeilenweise gescrollt werden?

Um in einem RecList-Objekt prozedural auf- und abzuscrollen wird mittels der Funktion RecRead() und den Optionen \_RecPrev oder \_RecNext der vorhergehende oder nächste Datensatz gelesen. Damit der gelesene Datensatz in der Liste selektiert wird, muss das RecList-Objekt mit folgender Anweisung aktualisiert werden:

### Beispiel:

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

Bei Umsatzzahlen soll die Währungsbezeichnung "EUR" dargestellt werden. Wie kann das erreicht werden?

Zur Anzeige einer Währungsbezeichnung sind folgende Schritte notwendig:

- Aufnahme eines Datenbankfeldes vom Typ alphanumerisch welches die Umsatzwerte darstellt.
- Setzen der Eigenschaft ClmAlign dieser Spalte auf den Wert \_WinClmAlignRight, damit der Inhalt der Spalte rechtsbündig dargestellt wird.
- Definition der Ereignisfunktion EvtLstDataInit, in der das alphanumerische Feld den Umsatzwert erhält und mit Darstellung der Währungsbezeichnung formatiert wird. Dazu wird die Konvertierungsfunktion CnvAF() mit der Option \_FmtNumCurrencyIntl zur Darstellung des Währungscodes verwendet.

### Beispiel:

```
sub EvtLstDataInit( aEvt : event; // Ereignis aID : bigint; // Datensatz-ID oder Zeilennummer)
```

Wie kann die Sortierung eines RecList-Objekts zur Laufzeit geändert werden?

Da die Sortierung eines RecList-Objekts über den in der Eigenschaft DbKeyNo angegebenen Schlüssel läuft, lässt sich die Sortierung der RecList jederzeit durch Setzen der Eigenschaft ändern. Die Sortierung kann auch durch Anklicken der entsprechenden Spaltenköpfe geändert werden, vorausgesetzt die Eigenschaften ClmSortFlags und DbKeyNo der entsprechenden Spalten ist gesetzt.

Wie kann ein bestimmter Datensatz im RecList-Objekt selektiert werden?

Mit der Option \_WinLstRecDoSelect kann ein bestimmter Datensatz im RecList-Objekt selektiert werden. Der Datensatz, welcher selektiert dargestellt werden soll, muss sich in den Feldpuffern befinden. Soll ein Datensatz mit einem bestimmten Schlüsselwert selektiert werden reicht es aus, dass der oder die Schlüsselfelder den entsprechenden

Wert zugewiesen bekommen. Voraussetzung hierbei ist, dass der Schlüssel mit dem in der Eigenschaft DbKeyNo angegebenen Schlüssel übereinstimmt. Ist das nicht der Fall, muss der Datensatz gelesen werden.

### Beispiel:

```
AdrNummer # 3;$RecList->WinUpdate(_WinUpdOn, _WinLstRecDoSelect);AdrOrt # 'Frankfurt';RecRead($R
```

Wie kann ermittelt werden, welcher Datensatz im RecList-Objekt selektiert ist? Über die Eigenschaft DbRecID bzw. DbRecID64 des RecList-Objekts kann zu jedem Zeitpunkt die Datensatz-ID des selektierten Datensatzes ermittelt werden. Über die Datensatz-ID kann der Datensatz gelesen werden.

### Beispiel:

```
RecRead($RecList->wpDbFileNo, 0, _RecID, $RecList->wpDbRecID64)
```

Können bestimmte Datensätze von der Anzeige ausgenommen werden? Für diesen Zweck gibt es das Ereignis EvtLstRecControl. Über den Rückgabewert der in diesem Ereignis angegebenen Funktion wird entschieden, ob ein Datensatz angezeigt wird oder nicht.

### Beispiel:

```
sub EvtLstRecControl( aEvt : event; // Ereignis aRecId : bigint; // Record-ID des Datensatzes
```

Im obigen Beispiel werden in dem RecList-Objekt nur Datensätze angezeigt, die im Feld "Ort" die Bezeichnung "Frankfurt" haben.

Aus Performancegründen sollte dieses Ereignis nur dann verwendet werden, wenn die Anzahl der Sätze die angezeigt werden, überwiegt und keine allzu großen Lücken vorhanden sind, da jeder Satz der übersprungen wird auch gelesen werden muss. Alternativ kann die Anzeige auch auf Basis einer Selektion (Eigenschaft DbSelection) oder eines Datensatzfilters (Eigenschaft DbFilter) durchgeführt werden.

Können innerhalb eines RecList-Objekts Felder aus unterschiedlichen Dateien dargestellt werden?

Dies ist möglich. Dazu müssen in dem Ereignis EvtLstDataInit des RecList-Objekts die Feldpuffer der betreffenden Felder, beispielsweise durch Lesen des gewünschten Datensatzes, gefüllt werden.

Beim Scrollen im RecList-Objekt soll der jeweils selektierte Datensatz in den Eingabeobjekten dargestellt werden.

Um zu erreichen, dass die Eingabeobjekte aktualisiert werden, muss in dem Ereignis EvtLstSelect des RecList-Objekts eine Funktion definiert werden, die ein Update auf die Felder durchführt.

### Beispiel:

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

Das Ereignis EvtLstSelect wird immer dann ausgelöst, wenn sich die selektierte Zeile in einer RecList ändert.

Standardmäßig werden in dem Ereignis EvtLstSelect lediglich die Feldpuffer der im RecList-Objekt angezeigten Datenbankfelder gefüllt. Ist es erforderlich, dass auch Felder der Datei welche nicht in dem Objekt vorhanden sind aktualisiert werden, muss der Datensatz in dem Ereignis EvtLstSelect gelesen werden.

### Beispiel:

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

Im obigen Beispiel wird der Datensatz über die interne Datensatz-ID gelesen. Ist der Schlüssel, über den das RecList-Objekt sortiert ist (DbKeyNo) ein eindeutiger Schlüssel und wird das Schlüsselfeld auch in der Liste angezeigt, kann der Datensatz auch über diesen Schlüssel gelesen werden.

```
RecRead(fKnd, aEvt:Obj->wpDbKeyNo, 0);
```

In einem Dialog werden Datensätze erfasst. Wie kann erreicht werden, dass ein neu angelegter Datensatz in einem RecList-Objekt des gleichen Dialoges angezeigt und selektiert dargestellt wird?

Damit der neue Datensatz in dem RecList-Objekt dargestellt werden kann, muss RecList neu aufgebaut werden. Mit der Funktion WinUpdate() und der Option WinUpdOn wird ein Update auf das RecList-Objekt durchgeführt. Um den neuen Datensatz zu selektieren, muss noch zusätzlich die Option WinLstRecDoSelect verwendet werden.

### Beispiel:

```
$RecList->WinUpdate(_WinUpdOn, _WinLstRecDoSelect)
```

Die Option WinLstRecDoSelect bewirkt, dass der Datensatz welcher in den Feldpuffern steht, im RecList-Objekt selektiert wird.

Wenn das RecList-Objekt angeklickt wird, werden anschließend die Eingabeobjekte mit den Daten des selektierten Datensatzes überschrieben. Wie kann dies verhindert werden?

Standardmäßig wird, wenn das RecList-Objekt den Fokus verliert, der selektierte Datensatz in die Feldpuffer übertragen. Ist dieses Verhalten nicht erwünscht, muss bei der Eigenschaft LstFlags des RecList-Objektes die Option WinLstRecFocusTermReset entfernt werden. Ist das Flag nicht gesetzt, werden Änderungen der Feldpuffer, bedingt durch das RecList-Objekt, wieder verworfen.

Nach dem Aufruf eines Frames mit einem RecList-Objekt sind Eingabeobjekte mit den Daten des selektierten Satzes bereits vorbelegt. Diese Felder sollen aber leer ein. Da der selektierte Datensatz vom RecList-Objekt gelesen wurde, ist der Datensatzpuffer gefüllt.

Um zu erreichen, dass nach dem Aufruf des Frames die Felder leer sind, muss bei der RecList in der Eigenschaft DbRecBuf die Datei angehakt werden, die in der



Eigenschaft DbFileNo angegeben ist. Vor der Anzeige des Frames muss der Datensatzpuffer mit der Funktion RecBufClear() geleert werden.

Durch Verwendung von DbRecBuf werden die Feldpuffer vor dem Aufbau der RecList gesichert und anschließend wieder hergestellt.

Können in einem RecList-Objekt Bilder angezeigt werden?

In einem RecList-Objekt können keine Bilder angezeigt werden.

Die Ausnahme bilden Icons. Werden in einer Spalte der Liste ganzzahlige Werte (Ganzzahlig kurz, Ganzzahlig lang oder Ganzzahlig 64) angezeigt, kann statt der Zahl ein Icon aus einem Symbolsatz (Tile) angezeigt werden. Die Zahl entspricht der Position des Symbols innerhalb des Tile.

Wenn ein Symbol angezeigt werden soll, muss die Eigenschaft ClmTypeImage auf \_WinClmTypeImageC16 oder \_WinClmTypeImageUser gesetzt werden. Bei \_WinClmTypeImageC16 wird das CONZEPT 16 Tile verwendet. Im anderen Fall wird das benutzerdefinierter Tile verwendet, das beim RecList-Objekt in der Eigenschaft TileNameUser angegeben wurde.

Wie kann verhindert werden, dass der Anwender die Reihenfolge der Spalten verändert?

Wird der Eigenschaft LstStyle der Wert \_WinLstHeader zugewiesen, ist das Verändern der Reihenfolge nicht mehr möglich. Eine Änderung der Spaltenbreite kann noch vorgenommen werden.

Der Anwender ändert in der Applikation die Reihenfolge der Spalten. Wie kann erreicht werden, dass diese Änderung bei Verlassen des Dialoges bzw. der Applikation erhalten bleibt?

Die Position einer Spalte kann über die Eigenschaft ClmOrder ermittelt und gesetzt werden. Um die Aufgabenstellung zu lösen, wird in der Ereignisfunktion EvtClose die Position aller Spalten des RecList-Objekts ermittelt und in eine Datei geschrieben.

### **Beispiel:**

```
sub EvtClose( aEvt : event; // Ereignis) : logic;local{ tObj : int;}{ // Abarbeiten der Spalten
```

Damit über die Funktion WinInfo() die Spalten in der Reihenfolge ihrer Anzeige ermittelt werden können, muss die Eigenschaft OrderPass des RecList-Objekts auf den Wert \_WinOrderShow gesetzt werden.

Beim Laden des Fensters wird in der Ereignisfunktion EvtInit die Datei gelesen und die Eigenschaft ClmOrder gesetzt.

Im RecList-Objekt werden vertikale und horizontale Linien gezeichnet. Wie können diese Linien ausgeblendet werden?

Um die Linien auszublenden muss der Eigenschaft ColGrid der Wert \_WinColUndefined zugewiesen werden.

## FAQ - Programmierung

### Häufige Fragen zur Programmierung

Wofür stehen die verschiedenen Abkürzungen bei den Befehlsnamen?



Alle Befehle sind über verschiedene Präfixe aufgebaut. Dabei stehen die Abkürzungen für verschiedene Befehlsbereiche. Ebenso werden Eigenschaften in verschiedene Klassen (Oberflächen-, Druck-, System-Eigenschaften usw.) eingeteilt.



### **Abkürzung Bedeutung**

Bin	Binary Object (Binäres Objekt, BLOb)
Cell	GanttGraph Cell (Zelle eines GanttGraph-Objektes)
Clm	Column (Eigenschaften von Spalten-Objekten)
Cnv	Convert (Konvertierungsbefehle)
Col	Color (Farbeeigenschaften und -konstanten)
Com	Component Object Model (Befehle und Eigenschaften der COM-Schnittstelle)
Cte	Container Element (Dynamische Strukturen)
Ctx	Container Element Extended (Oberflächenobjekt, welches COM-Objekte externer Applikationen, wie beispielsweise ein Word-Dokument, darstellen kann)
Date	Date (Datumseigenschaften und -befehle)
Db	Database (Datenbank-Befehle)
Db	Database (Eigenschaften der Datenstruktur)
Dbg	Debugger (Befehle und Konstanten des externen Debuggers)
Dde	Dynamic Data Exchange (Befehle der DDE-Schnittstelle)
Dll	Dynamic Link Library (Befehle zum dynamischen Einbinden von Bibliotheken)
Dta	Data Transaction (Transaktionsbefehle)
Err	Error (Fehlerbehandlung)
Evt	Event (Ereignis)
File	File (Datei / Tabelle)
Fld	Field (Datensatzfeld-Befehle)
Flt	Filter (Filter-Befehle)
Fmt	Format (Formatierungskonstanten und -eigenschaften)
Fsi	File System Interface (Befehle für externe Dateien)
Hdl	Handle (Deskriptor-Befehle)
Key	Key (Schlüssel / Index)
Lcl	Locale (Länderspezifische Einstellungen)
Link	Link (Verknüpfung)
Mail	E-Mail-Befehle
Msx	Message-Exchange-Befehle
Nti	Network information (Informationen über das Netzwerk)
pp	Print Property (Druck-Eigenschaft)
Ppv	Print Preview (Objekte der Druckvorschau)
Prt	Printer (Drucker oder Druck-Job)

_PrtProp	Print Property (Druck-Eigenschaft)
_r	Fehlerkonstanten der Datensatz-Befehle
Rec	Record (Datensatz-Befehl)
RecBuf	Record-Buffer (Datensatzpuffer)
Rtf	Rich Text Format (Objekte und Befehle der Rtf-Objekte)
Sbr	Subrecord (Teildatensatz)
Sck	Socket (Befehle und Konstanten der Sockel-Befehle)
Sel	Selection (Befehle und Konstanten der Selektionen)
sp	System Property (System-Eigenschaft)
Sto	Storage (Befehle und Konstanten für Storageobjekte)
Str	String (Befehle und Konstanten für Zeichenkettenbefehle)
Sys	System (System-Befehl)
_SysProp	System Property (System-Eigenschaft)
Tapi	Telephon Application Programming Interface (Telefon-Schnittstelle)
Text	Text (Befehle und Konstanten zur Bearbeitung interner Texte)
Type	Type Information (Information über Feld- und Variablentypen)
Urm	User and Rights Management (Benutzer- und Rechte-Verwaltung)
User	User (Benutzerinformationen)
vm	Variable method (Methoden von Variablentypen)
vp	Variable property (Eigenschaften von Variablentypen)
Win	Windows (Oberflächen)
WinMsd	Multi Selection Data (Mehrfachselektion)
_WinProp	Windows Property (Oberflächen-Eigenschaft)
wp	Windows Property (Oberflächen-Eigenschaft)

Was ist bei der Verwendung eines RtfEdit-Objekts in einem Notizbuch zu beachten?

Die Verwendung der Tastenkombination  +  führt bei dem RtfEdit-Objekt zum Anspringen des nächsten Tabulators. Befindet sich das RtfEdit-Objekt auf einer Notizbuchseite findet statt dessen ein Wechsel der Notizbuchseite statt.

Die gleiche Problematik ist auch vorhanden, wenn beim RtfEdit-Objekt in der Eigenschaft RtfEditFlags die Ausprägung \_WinRtfEditTabSingle gesetzt ist. Die Tastenkombination  +  soll dann zum Fokuswechsel aus dem RtfEdit-Objekt führen, es findet aber ein Wechsel der Notizbuchseite statt.

Das Anspringen des nächsten Tabulators oder der Fokuswechsel kann durch das Setzen der Eigenschaft Flags (des Notebook-Objektes) auf den Wert \_WinFlagNbCtrlTabOff erreicht werden.

### Beispiel:

```
sub EvtPageSelect( aEvt          : event; // Ereignis aPage          : int;    // Notebook-Seite aSel
```

Wie findet prinzipiell die Verarbeitung von Ereignissen statt?

In der Verarbeitung von Dialogen sind in CONZEPT 16 drei unterschiedliche Ebenen zu betrachten. Das Betriebssystem, das CONZEPT 16-System und die

Applikationsoberfläche.

Wird eine CONZEPT 16-Applikation gestartet, wird in der Regel sofort ein Dialog geöffnet. Innerhalb von CONZEPT 16 wird zu diesem Zeitpunkt keine Prozedur mehr verarbeitet. Alle weiteren Aufrufe von Funktionen werden von "Außen" angeregt.

Von "Außen" angeregt bedeutet, dass Ereignisse entweder durch das System oder den Benutzer ausgelöst werden. Damit solche Ereignisse verarbeitet werden können, läuft unter Windows eine Endlos-Schleife. Diese Ereignisschleife hat nur die Aufgabe Ereignisse entgegen zu nehmen und die Weiterverarbeitung anzustoßen.

Wird durch den Benutzer zum Beispiel mit dem Mauszeiger auf eine Schaltfläche geklickt, wird eine ganze Reihe von Tätigkeiten durchgeführt. Zunächst wird das Ereignis durch die Ereignisschleife des Betriebssystems festgestellt und in eine Warteschlange einsortiert. Die Ereignisse in dieser Warteschlange werden nacheinander abgearbeitet. Vom Betriebssystem wird der Typ des Ereignisses ermittelt. Einige Ereignisse werden vom Betriebssystem selbst verarbeitet (zum Beispiel das Verschieben eines Fensters, das Bewegen des Mauszeigers u.ä.), andere werden an das entsprechende Programm weitergereicht. In unserem Beispiel werden weitere Informationen gesammelt (welches Objekt angeklickt wurde) und das Ereignis an CONZEPT 16 weitergereicht.

Innerhalb von CONZEPT 16 wird geprüft, ob dem Ereignis eine Funktion zugeordnet wurde. Hat der Programmierer in dem Ereignis EvtClicked der Schaltfläche eine Funktion angegeben, wird diese aufgerufen. Nach dem Abarbeiten der Funktion wird dessen Ergebnis von CONZEPT 16 an das Betriebssystem weitergegeben. Im Falle von EvtClicked hat das Ergebnis keine Auswirkungen.

Das Ereignis ist damit abgearbeitet und das Betriebssystem kann das nächste Ereignis verarbeiten oder auf das nächste Ereignis warten.

Beim Testen des Dialogs in der Entwicklungsumgebung hat das Drücken von Schaltflächen und das Aufrufen von Menüeinträgen keine Wirkung.

In der Entwicklungsumgebung gibt es zwei Möglichkeiten einen Dialog zu testen. Über den Menüpunkt Datei / Test kann nur der Dialog getestet werden. Es werden keine Ereignisfunktionen aufgerufen. Damit bleibt das Drücken von Schaltflächen und das Aufrufen von Menüeinträgen ohne Funktion.

Im Editor kann über den Menüpunkt Prozedur / Ausführen eine Prozedur gestartet werden. Zeigt diese Prozedur einen Dialog an, werden auch alle Ereignisfunktionen, die über diesen Dialog aufgerufen werden, durchgeführt. Der Dialog kann dann auch in seiner Funktion getestet werden.

Wie kann nach dem Starten des Clients ein Bild als Hintergrund für die Applikation angezeigt werden?

Ab der CONZEPT 16-Version 4.4.05 kann ein Applikationsfenster mit einem Hintergrundbild versehen werden. Das Bild muss zuvor in die Datenbank importiert und kann dann über die Eigenschaft PictureName gesetzt werden.

Ab der Version 4.7.04 kann jedem Frame-Objekt ein Hintergrundbild zugeordnet werden. Dieses Bild kann sowohl in der Datenbank als auch als externe Datei

vorliegen.

Wie können mehrdimensionale Arrays definiert werden?

Ab der CONZEPT 16-Version 4.2 können in A+-Prozeduren mehrdimensionale Arrays angelegt werden, wenn man eine einfache Funktion zur Berechnung eines skalaren Index verwendet.

Ein Vektor kann wie folgt deklariert werden:

```
local{ vector : int[3];}
```

Die Variable vector besteht aus drei Variablen vom Typ int. Das dritte Element dieses Arrays kann z. B. mit vector[3] angesprochen werden. Die Nummerierung startet bei 1.

Bei mehrdimensionalen Arrays wird die weitere Dimension mit Hilfe der Multiplikation in der Klammer angegeben:

```
local{ arr2dim : int[4*3];}
```

Hierdurch werden zwölf Variablen angelegt. Das Element (x,y) aus einem Array kann über folgende Formel angesprochen werden:

$$(y - 1) * xdim + x$$

wobei **xdim** die Anzahl der Elemente pro Zeile ist.

y \ x	1	2	3	4
1	1 ((1 - 1) * 4 + 1)	2 ((1 - 1) * 4 + 2)	3 ((1 - 1) * 4 + 3)	4 ((1 - 1) * 4 + 4)
2	5 ((2 - 1) * 4 + 1)	6 ((2 - 1) * 4 + 2)	7 ((2 - 1) * 4 + 3)	8 ((2 - 1) * 4 + 4)
3	9 ((3 - 1) * 4 + 1)	10 ((3 - 1) * 4 + 2)	11 ((3 - 1) * 4 + 3)	12 ((3 - 1) * 4 + 4)

Auf diese Weise lassen sich beliebig-dimensionale Arrays erstellen.

Die Formel zur Berechnung der Position des Elements (x,y,z) in einem dreidimensionalen Array ermittelt sich aus

$$x + (y - 1) * xdim + (z - 1) * ydim * xdim$$

Mehrdimensionale Arrays können in der Auswertung verschiedener Daten verwendet werden. Beispielsweise können Umsatzzahlen nach Gebieten, Warengruppen und Monaten aufgeschlüsselt werden.

Wie kann das aktuelle Verzeichnis gewechselt werden?

In CONZEPT 16 kann mit den A+-Prozeduren das aktuelle Verzeichnis gewechselt werden. Dies geschieht mit dem Befehl FsiPathChange(). Das Wechseln des aktuellen Verzeichnisses ist bei der Ausführung durch den CONZEPT 16-Client verhältnismäßig unkritisch. Bei der Ausführung des Befehls durch den CONZEPT 16-Server, der externen Programmierschnittstelle oder der Web-Schnittstelle kann es allerdings zu Überschneidungen kommen, da mehrere Prozeduren in einer Prozess-Umgebung ablaufen. Das äußert sich derart, dass externe Dateien, die sich im aktuellen Pfad befinden sollen, nicht gefunden werden. Dies hängt mit dem Wechsel des aktuellen

Verzeichnisses zusammen.

Das aktuelle Verzeichnis ist Bestandteil der Prozess-Umgebung. Laufen jetzt mehrere CONZEPT 16-Prozeduren in einer Prozess-Umgebung ab, z. B. mehrere Prozeduren werden auf dem CONZEPT 16-Server mit dem Befehl RmtCall() gestartet, kann die Prozess-Umgebung durch eine Prozedur verändert werden. D. h. eine Prozedur verändert die Umgebung für alle Prozeduren.

Die Veränderung des aktuellen Verzeichnisses sollte vermieden werden, gerade, wenn die Prozeduren vom Server, der externen Programmierschnittstelle oder der Web-Schnittstelle ausgeführt werden sollen. Alle Prozedurbefehle zum Öffnen (FsiOpen()), Löschen (FsiDelete()) oder Umbenennen (FsiRename()) von Dateien bzw. zum Lesen von Verzeichnissen (FsiDirOpen()) können einen vollständigen Pfad beinhalten, womit die Verwendung des aktuellen Verzeichnisses nicht notwendig ist.

Können aus A- Prozeduren Funktionen von A+ Prozeduren aufgerufen werden? Auch aus A- Prozeduren können Funktionen von A+ Prozeduren aufgerufen werden. Dazu muss folgender Befehl angegeben werden:

```
Call('Lib.Rec:Insert', ...);
```

Der Call()-Befehl ruft die Funktion Insert aus der Prozedur Lib.Rec auf.

Warum werden die Prozeduraufrufe von A- Prozeduren aus A+ Prozeduren mit Call() ignoriert?

Nach der Übersetzung liegen die Prozeduren in einem P-Code vor. Der P-Code unterscheidet sich je nach verwendetem Übersetzer. Beim Übersetzen von A-Prozeduren entsteht ein anderer P-Code, als beim Übersetzen von A+ Prozeduren.

Zur Interpretation zur Laufzeit werden daher auch zwei unterschiedliche Laufzeitumgebungen benötigt. Um den CONZEPT 16-Server, die Web-Schnittstelle und die Windows Programmierschnittstelle möglichst schlank zu halten, verfügen diese nur über die Laufzeitumgebung für A+ Prozeduren. A- Prozeduren können somit nicht interpretiert werden. Der Aufruf von diesen Prozeduren wird daher ignoriert.

Damit die entsprechenden Funktionalitäten ausgeführt werden können, müssen diese in A+ Prozeduren überführt werden. A+ Prozeduren können von A- Prozeduren aufgerufen werden (siehe "Können aus A- Prozeduren Funktionen von A+ Prozeduren aufgerufen werden?").

Beim Abfragen der Eigenschaft BiasMinutes kommt immer 0 zurück.

In der Eigenschaft vpBiasMinutes ist die Verschiebung der Uhrzeit durch die verwendete Zeitzone abgelegt. Die Eigenschaft gibt nur dann die korrekte Zeitzone zurück, wenn zuvor die Methode vmSystemTime() der Variablen aufgerufen wurde. Wurde die Methode nicht aufgerufen, wird 0 zurückgegeben.

Wozu werden Deskriptoren benötigt?

Deskriptoren bilden eine einfache Möglichkeit auf ein bestimmtes Objekt zuzugreifen. Bei dem Objekt kann es sich um ein Dialog-Objekt (Frame, Eingabe-Objekt, Menüeintrag usw.) oder um Puffer handeln. Mit Hilfe eines Deskriptors eines Dialog-Objektes können die Eigenschaften des Objektes manipuliert werden:

## Kontakt

```
hdlObj->wpCaption # 'Name';
```

Der Deskriptor eines Objektes kann mit der Funktion \$ und dem Namen des Objektes ermittelt werden. Bei der Verwendung von Ereignisfunktionen wird der Deskriptor des auslösenden Objekts bereits im Übergabeparameter aEvt:Obj übergeben.

Handelt es sich um ein Puffer-Objekt (zum Beispiel einen internen Text, einen Selektionspuffer oder einen Puffer einer externen Datei), kann der entsprechende Puffer manipuliert oder an einen Befehl übergeben werden.

```
// Ersten Datensatz aus einer Selektion lesenRecRead(CST.F.Customer, hdlSel, _RecFirst);
```

Weitere Möglichkeiten ergeben sich aus der Beschreibung der entsprechenden Befehle.

Worin besteht der Unterschied zwischen dem Zugriff auf ein Dialog-Objekt über den Namen und seinem Deskriptor?

Bei der Verwendung der Funktion \$ zusammen mit dem Namen des Objekts, wird der Objektbaum nach einem Objekt mit dem entsprechenden Namen durchsucht. Sobald der Name gefunden wurde, wird der Aufruf durch den Deskriptor des Objekts ersetzt.

Wird gleich der Deskriptor verwendet, entfällt die Suche. Die Funktion wird schneller abgearbeitet.

Wenn möglich sollte immer über einen Deskriptor zugegriffen werden. In der Regel kann ein Eltern-Objekt schneller mit dem Befehl WinInfo(), als über seinen Namen ermittelt werden. Ebenso ist einer Schleife, welche die Deskriptoren aller Objekte eines übergeordneten Objektes ermittelt, gegenüber einer Auflistung aller Namen, den Vorzug zu geben.

Müssen bei einem Funktionsaufruf immer alle Parameter angegeben werden?

In A+ Prozeduren können Parameter von Funktionen als optionale Parameter angegeben werden. In der Deklaration des Funktionskopfes wird dem Parameter opt vorangestellt.

### Beispiel:

```
sub Add4( aValue1 : int; aValue2 : int; opt aValue3 : int; opt aValue4 : int;) : int;{ return
```

Die Funktion Add4() kann mit zwei, drei oder vier Parametern aufgerufen werden. Zurückgegeben wird die Summe der übergebenen Parameter. Parameter, die nicht angegeben werden, sind in der Funktion mit 0 initialisiert.

Für die Übergabe einer variablen Anzahl von Parametern können dynamische Strukturen verwendet werden. Die Parameter werden in einer Liste oder einem Baum gespeichert und die Wurzel der Struktur als Parameter an die Funktion übergeben. Innerhalb der Funktion müssen dann die Parameter aus der Struktur ermittelt werden.

In einer Mehrbenutzerumgebung soll eine Selektion mehrfach gestartet werden können.

## Kontakt

Die Selektion wird vor dem Start kopiert (SelCopy()). Der neue Name setzt sich aus einem Kürzel für die Selektion und der User-ID des Benutzers (UserInfo()) zusammen.

```
// Name der Selektion erzeugtSelNew # 'TMP_SEL_' + UserInfo(_UserNumber);// Selektion kopierenS
```

Die bei einer Selektion angehängte Prozedur wird nicht ausgeführt.

Bei der Definition einer Selektion kann hinter jeder Abfrage eine Prozedur angegeben werden, die nach Überprüfung der Abfrage durchgeführt wird.

Die dort angegebene Prozedur kann entweder eine 4.0 kompatible Prozedur (A-) oder eine A+ Prozedur sein. In Abhängigkeit von der Umgebung, in der die Selektion gestartet wird, kann aber nur eine A- oder eine A+ Prozedur durchgeführt werden, da die Rückgabe, ob ein Datensatz in die Selektionsmenge aufgenommen werden soll oder nicht, über unterschiedliche Rückgabemechanismen realisiert werden musste.

Dies hat zur Folge, dass eine angegebene A+ Prozedur nur dann durchgeführt wird, wenn die Selektion auch von einer A+ Prozedur (mit dem Befehl SelRun()) gestartet wurde. Erfolgte der Selektionsstart über eine A- Prozedur oder dem Menüpunkt **Bearbeiten / Selektieren / Selektion durchführen...**, wird eine angehängte A+ Prozedur nicht ausgeführt.

Bei den A+ Prozeduren gibt es die Funktion setResult() nicht. Wie kann die Prozedur nach Abfrage bei Selektionen realisiert werden?

Bei den A- Befehlen konnte mit der Funktion setResult() in der "Prozedur nach Abfrage" bestimmt werden, ob ein Datensatz in die Selektionsmenge aufgenommen wird. Bei Verwendung einer A+ Prozedur als "Prozedur nach Abfrage" geschieht dies über den Rückgabewert der main-Funktion. Zu diesem Zweck wird eine main-Funktion mit einer Rückgabe vom Typ logic deklariert. Beim Beenden der Funktion mit true wird der Datensatz in die Selektionsmenge aufgenommen, bei false nicht.

Wie kann eine Selektionsmenge nach Werten sortiert werden, die sich erst zur Laufzeit der Selektion ergeben?

In der Ausgangsdatei der Selektion wird ein Hilfsfeld angelegt, welches während des Selektionslaufes die berechneten Werte aufnimmt. Die Sortierung der Selektion wird über dieses Feld vorgenommen. In der "Prozedur nach Abfrage" wird der berechnete Wert dem Hilfsfeld zugewiesen.

Da bei dem Zugriff auf eine Selektionsmenge der Datensatzinhalt als Grundlage für die Positionierung dient, muss der alternative Selektionsmodus vor dem Zugriff auf die Selektionsmenge eingeschaltet werden. Der alternative Modus wird mit der Funktion SelRead() in Verbindung mit der Option SelKeyMode aktiviert.

Wie kann die Laufzeit von Selektionen verkürzt werden?

Siehe dazu den Abschnitt Selektionen optimieren.

Welchen Zweck erfüllt die Ausnahmebehandlung?

Bei vielen Anweisungen wird nach der Anweisung geprüft, ob ein zulässiges Resultat vorhanden ist. In Abhängigkeit davon werden dann entweder weitere Anweisungen durchgeführt oder das unzulässige Resultat wird in irgendeiner Form behandelt.



Diese Vorgehensweise kann Anweisungsfolgen stark verkomplizieren und auch unübersichtlicher machen. Durch die Verwendung der Ausnahmebehandlung in Form des sogenannten try-Blocks

kann dies vermieden werden. Dabei führen Verarbeitungsfehler zum Verlassen des try-Blocks. Der jeweilige Verarbeitungsfehler kann dann nach dem try-Block behandelt werden.>

### Beispiel:

```
try{ RecRead(1, 1, _RecLock);    // Kunde lesen und sperren RecLink(2, 1, 1, _RecLock); // Auf
```

Werden innerhalb des try-Blocks Funktionen aufgerufen, kann anstelle von try auch trysub verwendet werden.

Wie kann bei größeren try-Blöcken die Fehlerposition ermittelt werden?

Bei größeren try-Blöcken reicht der Fehlerwert vielfach zur anschließenden Behandlung nicht aus, da derselbe Fehlerwert an unterschiedlichen Stellen innerhalb des try-Blocks auftreten kann.

Zur Feststellung der Fehlerposition können innerhalb des try-Blocks entsprechende Markierungen, sogenannte LABELS verwendet werden.

Ein Label beginnt mit einem Doppelpunkt, gefolgt von einem Namen und bezeichnet einen bestimmten Abschnitt im try-Block, nämlich von der Anweisung nach dem Label bis zum nächsten Label.

In der Fehlerbehandlung kann dann mit der Funktion ErrPos() die Position des aufgetretenen Fehlers bestimmt werden.

### Beispiel:

```
try{ :Customer CST.iNumber # 23; RecRead(CST.F.Customer, 1, _RecLock); ... :Order ORD.iNum
```

ErrPos() liefert ein ganzzahliges Resultat, welches einem der definierten Labels entspricht.

Ist ErrPos() gleich 0, trat der Fehler vor dem ersten Label auf.

Wie können auftretende Laufzeitfehler abgefangen werden?

Ab der CONZEPT 16-Version 4.2 steht dem Programmierer eine Ausnahmebehandlung (try-Anweisung) zur Verfügung.

Laufzeitfehler führen normalerweise zu einem Abbruch der Prozedur. Innerhalb von try-Blöcken können Laufzeitfehler abgefangen werden und wie andere Fehlerwerte in der Fehlerbehandlung verarbeitet werden. Dazu muss allerdings für jeden einzelnen Typ von Laufzeitfehler das Abfangen in try-Blöcken explizit ein- und ausgeschaltet werden.

Dies geschieht durch den Befehl ErrTryCatch(). Soll beispielsweise eine Division durch Null abgefangen werden, lautet die Anweisung

ErrTryCatch( \_ErrDivisionByZero, true).

Die Einstellungen zum Abfangen von Laufzeitfehlern sind global. Damit sind Veränderungen für jeden try-Block relevant.



Laufzeitfehler, die außerhalb eines try-Blocks auftreten, führen immer zum Prozedurabbruch.

Wie kann erreicht werden, dass bei einem Fehler innerhalb eines try-Blocks dieser nicht verlassen wird?

In manchen Fällen muss ein Fehler direkt bei der Anweisung behandelt werden, da der Fehler Bestandteil der normalen Verarbeitung ist und nach der Behandlung die Verarbeitung der Anweisungsfolge fortgesetzt wird.

Zu diesem Zweck darf bei bestimmten Fehlerwerten der try-Block nicht verlassen werden (dies gilt vor allem bei Resultaten von Satzoperationen).

Mit dem Befehl ErrTryIgnore(int, int) kann veranlasst werden, dass der try-Block bestimmte Fehlerwerte einfach ignoriert.

Bei ErrTryIgnore() kann entweder ein einzelner Fehlerwert oder ein Bereich von Fehlerwerten angegeben werden.

Dabei wird bei jedem Aufruf von ErrTryIgnore() der zu ignorierende Fehlerwert neu gesetzt, d. h. bei mehreren ErrTryIgnore() summieren sich die Fehlerwerte nicht.

### **Beispiele:**

```
// Resultat _rLocked wird ignoriertErrTryIgnore(_rLocked);// Alle Resultate von RecRead werden vo
```

Die Anweisung ErrTryIgnore() muss immer innerhalb eines try-Blocks stehen.

Beim Laden eines Dialogs können die Feldpuffer nicht geleert werden.

In einem aufgerufenen Dialog sollen leere Eingabeobjekte dargestellt werden. Die dazugehörigen Felder werden in dem Ereignis EvtInit mit dem Befehl RecBufClear() gelöscht. Trotzdem sind die Eingabeobjekte mit Werten gefüllt.

Dieses Verhalten tritt nur auf, wenn vor dem Aufrufen des Dialoges der Eingabefokus auf einem RecList-Objekt oder einem Eingabe-Objekt steht. Der Eingabefokus kann erst dann in den neuen Dialog transferiert werden, wenn er angezeigt wird. Zu diesem Zeitpunkt ist das Ereignis EvtInit bereits beendet. Standardmäßig wird der selektierte Datensatz eines RecList-Objekts bzw. eines Eingabe-Objekts in die Feldpuffer übertragen, wenn der Fokus das Objekt verlässt.

Das Übertragen des selektierten Datensatzes in die Feldpuffer kann unterbunden werden, indem in der Eigenschaft LstFlags die Ausprägung WinLstRecFocusTermReset entfernt wird.

Das Leeren der Feldpuffer kann ebenfalls in dem Ereignis EvtCreated erfolgen. Zu diesem Zeitpunkt ist der Fokuswechsel bereits erfolgt.

## Kontakt

Das Verhalten tritt nicht auf, wenn der auf dem gleichen Rechner laufende externe Debugger die Verarbeitung des Ereignisses EvtInit unterbricht. In diesem Fall wechselt der Fokus von dem RecList-Objekt zum Debugger und erst dann wird das Ereignis EvtInit durchgeführt. Das Löschen des Feldinhaltes erfolgt also nach dem Fokuswechsel.

Wie können Funktionsaufrufe in der Programmierung verfolgt werden?  
Der externe Debugger verfügt über einen Protokollbereich. Hier werden alle Meldungen, die mit dem Befehl DbgTrace() ausgegeben werden, dargestellt. Im Prozedurbereich wird im Falle eines Laufzeitfehlers oder dem Befehl DbgControl(\_DbgStop) die angehaltene Prozedur angezeigt.

Im Protokollbereich kann neben den Trace-Meldungen, die Ein- und Aussprünge aus einer Funktion protokolliert werden. Das Einschalten der Protokollierung kann entweder im externen Debugger mit den Menüpunkten **Protokoll / Aufrufe extern**, **Protokoll / Aufrufe intern** bzw. **Protokoll / Funktionsaustritt** oder mit dem Befehl DbgControl(\_DbgEnter) bzw. DbgControl(\_DbgLeave) erfolgen.

Nicht alle in einer Prozedur deklarierten Variablen werden in der Variablenliste des externen Debuggers angezeigt.

Um eine Übersichtlichkeit zu gewährleisten, werden nur Variablen in die Liste aufgenommen, die in der Prozedur angesprochen werden.

Wie können aus CONZEPT 16 heraus registrierte Dateien gestartet werden?  
Über den Befehl SysExecute() lassen sich Dokumente mit registrierten Dateierweiterungen aufrufen. In diesem Fall wird in (alpha1) ein Stern (\*) gefolgt vom Namen des Dokumentes angegeben.

```
SysExecute('*D:\Doc\Readme.doc', '', 0); // Starten eines Word-Dokumentes
```

Wie können aus CONZEPT 16 heraus Befehle der Betriebssystemshell aufgerufen werden?

Über den Befehl SysExecute() lassen sich Kommandos der Shell ausführen. Dazu muss in (alpha1) von SysExecute() der Windows-Befehlsprozessor durch die Angabe von 'cmd' gestartet werden. In (alpha2) wird der Shell-Befehl inklusive Parameter angegeben. Die möglichen Parameter des Befehlsprozessors können mit help cmd abgerufen werden.

```
SysExecute('cmd', '/c copy ' + _Sys->spPathTemp + '\a.dat ' + _Sys->spPathTemp + '\b.dat', 0); //
```

Wie kann aus CONZEPT 16 eine DFÜ-Verbindung gestartet werden.

Unter Windows-Betriebssystemen steht dafür das Programm Rasdial zur Verfügung.

```
SysExecute('Rasdial', '<Connection name>', 0);
```

Nach dem Mailversand aus CONZEPT 16 soll die Verbindung zum Mailserver automatisch beendet werden.

Unter Windows-Betriebssystemen kann bei einem installierten DFÜ-Netzwerk das Programm Rasdial verwendet werden. Dieses Programm kann mit verschiedenen Parametern aufgerufen werden, darunter auch Parameter zum Aufbau und zum Trennen von Verbindungen.

Ein Verbindungsabbau kann über den Befehl `SysExecute('rasdial', '-h <entry>', 0)` erfolgen. In diesem Fall erfolgt der Abbau im Hintergrund. Nähere Informationen zum Programm "Rasdial" kann der Hilfe zu Windows entnommen werden.

Wie kann auf einfache Weise ein zusammengesetzter Datentyp gelöscht werden? Einige der zusammengesetzten Datentypen (zum Beispiel font) besitzen eine ganze Reihe von Komponenten. Um eine Variable von diesem Typ zu leeren, müssen alle Komponenten auf 0 bzw. " gesetzt werden. Dies geschieht schneller mit der Konstanten NULL.

### Beispiel:

```
local {    cMoment : caltime;    rRect    : rect; }{ ... /* cMoment->vpYear # 0; cMoment->vp
```

Kann während der Darstellung eines Dialogs eine Funktion ausgeführt werden? Ja. Dazu wird ein asynchroner Dialog benötigt. Wird ein Dialog mit der Anweisung WinDialog() oder WinDialogRun() aufgerufen, bleibt die Verarbeitung der Funktion an dieser Stelle stehen, bis der Dialog wieder geschlossen wurde. Alle Funktionen werden in der Folge über Ereignisse aufgerufen.

Ein Dialog kann aber auch als asynchroner Dialog aufgerufen werden. Dazu wird bei der Anweisung WinDialogRun() der Parameter WinDialogAsync angegeben. Die Verarbeitung der Funktion wird dann sofort nach der Anzeige des Dialogs fortgesetzt. Nach dem Aufruf des Dialoges muss also die Verarbeitung erfolgen.

Während der Verarbeitung kann mit dem Befehl WinDialogResult() ermittelt werden, ob der Dialog geschlossen wurde. In der Regel folgt nach der Anzeige des Dialoges eine while-Schleife, in der die Abbruchbedingung des Algorithmus und die Bedingung (`tHdlDialog->WinDialogResult() != _WinIdCancel`) abgeprüft wird. Das Resultat wird ebenfalls gesetzt, wenn ein Button-Objekt (`TypeButton = _WinBtnUserBreak`) gedrückt wird. In diesem Fall kann auch eine laufende Selektion unterbrochen werden. Bei einem anschließenden WinClose() wird entweder nach Beendigung des Algorithmus oder nach dem Drücken der Abbrechen-Schaltfläche der Dialog geschlossen.

Innerhalb der Schleife wird die gesamte Rechenzeit des Prozesses aufgebraucht, sodass eine Aktualisierung des Dialoges nicht möglich ist. Innerhalb der Schleife muss daher der Befehl WinSleep() stehen, um ein Update der Oberfläche zu ermöglichen.

Ein programmiertes Beispiel befindet sich in der Beispiele-Datenbank und im Abschnitt Schreiben von externen Dateien.

Wie können Informationen an eine beim Server ausgeführte Funktion übergeben werden?

Wird eine Funktion mit der Anweisung RmtCall() aufgerufen, wird die Funktion beim CONZEPT 16-Server ausgeführt. Der Server erzeugt dafür einen neuen Benutzer in der Datenbank. Dieser Benutzer hat seine eigenen Feldpuffer. Die notwendigen Informationen zur Durchführung der Funktion müssen entweder in den Parametern der Funktion übergeben, oder über die Möglichkeiten zum Informationsaustausch zwischen zwei Benutzern (siehe unten) übertragen werden.

Da die beim Server ausgeführte Funktion parallel zur Verarbeitung auf dem Client abläuft, muss eine Möglichkeit zur Synchronisation der beiden Prozesse geschaffen werden. In der einfachsten Form kann ein Datensatz angelegt und gesperrt werden, der zum Abschluss der Funktion wieder entsperrt wird.

Wie können Informationen zwischen zwei Benutzern in der Datenbank ausgetauscht werden?

Zum Informationsautausch zwischen zwei Benutzern stehen verschiedene Möglichkeiten zur Verfügung:

- **Austausch über externe Dateien**

Verfügen beide Clients über ein gemeinsames externes Verzeichnis mit Schreib- und Leserechten, können die entsprechenden Informationen über externe Dateien ausgetauscht werden. Dabei können unterschiedliche Formate verwendet werden (siehe Befehle für XML-Verarbeitung oder Message-Exchange-Befehle). Hat jeder Benutzer einen Client mit Oberfläche gestartet, kann das externe Verzeichnis mit einem EvtFsiMonitor überwacht werden. Das Abholen der Daten kann aber zu einem beliebigen Zeitpunkt erfolgen.

- **Austausch über Socket-Verbindungen**

In diesem Fall benötigen beide Benutzer ein Fenster, für den das Ereignis EvtSocket aufgerufen wird, wenn Daten übertragen werden sollen. Das Datenformat ist wie beim Austausch über externe Dateien frei wählbar.

- **Austausch über Datensätze**

Die Informationen werden in Datensätzen gespeichert. Dabei kann auch eine temporäre Datei angegeben werden, wenn die Informationen nicht über einen Neustart des Datenbankprozesses hinaus erhalten bleiben müssen. Der Empfänger der Daten muss in regelmäßigen Abständen überprüfen, ob Informationen für ihn vorliegen. Das Datenformat wird durch die Datenstruktur bestimmt.

- **Austausch über zentrale Datenobjekte**

Der Austausch kann ebenfalls über zentrale Datenobjekte erfolgen. Die Daten werden nicht gespeichert und spätestens beim Abmelden des Benutzers, der die Daten geschrieben hat, gelöscht. Die Datenmenge ist auf maximal 64 MB begrenzt. Es können nur Zeichenketten übertragen werden.

Wann ist es sinnvoll Informationen in dynamischen Objekten zu verarbeiten?

Die Verarbeitung in dynamischen Objekten ist immer dann sinnvoll, wenn sonst die Informationen wiederholt aus der Datenbank gelesen werden müssen und kein Zugriff von anderen Clients auf diesen Datenbestand notwendig ist.

Muss der Datenbestand in einer Baum-Struktur organisiert werden, bietet sich die Organisation mit CteNode-Objekten an.

Wie wird die grafische Erweiterung aktiviert?

## Kontakt

Die grafische Erweiterung ermöglicht Schnittstellen die Verwendung von Befehlen der Benutzeroberfläche und Druckfunktionen.

Beim SOA-Service kann die grafische Erweiterung aktiviert werden, indem in der Konfigurationsdatei der Eintrag c16\_proc\_extended auf Y gesetzt wird.

Bei der DLL-Schnittstelle wird die grafische Erweiterung aktiviert, wenn die c16\_pgxe.dll statt der c16\_pgxw.dll geladen wird.

FAQ - Verschiedenes

Häufige Fragen zu anderen Themen

Wo liegen die Unterschiede zwischen den Lizenztypen?

Es existieren nur noch vier unterschiedliche Lizenzeditionen mit folgendem Funktionsumfang:

- **Developer-Edition (CD...)**

Die Developer-Edition umfasst alle Schnittstellen und Funktionen. Es steht somit der Server (für bis zu 100 Benutzer), der Standard-Client, der Advanced-Client, der externe Debugger, die DLL-, ODBC-, Web- und PHP-Schnittstelle sowie der Druckertreiber zur Verfügung. Die Lizenz kann unter Windows- und Linux-Systemen mit 32 und 64 Bit-Architekturen mit beliebiger Anzahl von logischen Prozessoren betrieben werden. Zusätzlich ist in dieser Edition auch die Hot-Standby-Option enthalten.

- **Standard-Edition (CS...)**

Die Standard-Edition umfasst den Server (die Anzahl der Benutzer wird bei der Bestellung mit angegeben) und den Standard-Client. Die Lizenz wird zur Verwendung unter Linux oder Windows bestellt und kann auf 32 Bit-Architekturen mit bis zu vier logischen Prozessoren eingesetzt werden.

- **Advanced-Edition (CA...)**

Diese Edition umfasst den Server, den Standard-Client, den Advanced-Client, die DLL- und die Web-Schnittstelle. Der Server kann unter Windows oder Linux eingesetzt werden (Multilizenz). Die Lizenz ist für 32 Bit-Architekturen mit bis zu acht logischen Prozessoren ausgelegt.

- **Enterprise-Edition (CE...) / (TE...)**

Diese Edition umfasst alle Schnittstellen und Funktionen. Es steht der Server, der Standard-Client, der Advanced-Client, der externe Debugger, die DLL-, ODBC-, Web- und PHP-Schnittstelle sowie der Druckertreiber zur Verfügung. Die Lizenz kann unter Windows- und Linux-Systemen (Multilizenz) mit 32 und 64 Bit-Architekturen mit beliebiger Anzahl von logischen Prozessoren betrieben werden.



Standard-, Advanced- und Enterprise-Editionen können zusätzlich mit der Hot-Standby-Option erweitert werden.

Warum erscheint die Meldung "Temporäre Datei kann nicht angelegt werden"?

Neben der Meldung "Temporäre Datei kann nicht angelegt werden" wird noch ein Fehlerwert ausgegeben, der über die Ursache informiert.

Beispiele einiger möglicher Fehlerwerte unter Windows:

Code 3 Ziellaufwerk oder Pfad ist nicht vorhanden (Path not found)

Code 5 Zugriff verweigert (Access is denied)

Code 21 Ziellaufwerk ist nicht bereit (Error reading drive)

Code 32 Datei ist von einem anderen Prozess im Zugriff (Sharing violation)

Unter Windows NT kann die Bedeutung dieser Errorcodes in der Eingabeaufforderung mit dem Befehl `net helpmsg` ermittelt werden. (Hilfe hierzu erhält man mit `net`

helpmsg /?).

Siehe auch: Welche temporären Dateien werden vom CONZEPT 16-Client angelegt?  
und Welche temporären Dateien werden vom CONZEPT 16-Server angelegt?

Was ist beim Umstieg von den Versionen 4.0 und 4.2 auf die Version 4.7 oder 5.x zu beachten?

Die Version 4.7 und 5.x enthält alle Funktionalitäten der Versionen 4.0 und 4.2. An den Parametern, Prozeduren, Masken usw. müssen keine Änderungen vorgenommen werden.

Es ist lediglich zu beachten, dass nach der Umwandlung der Datenbank in die Version 4.7 oder 5.x, diese nicht mehr mit einem CONZEPT 16-Client der Version 4.0 oder 4.2 geöffnet werden kann.

Ein Eingriff in die Programmierung der Datenbank ist erst dann erforderlich, wenn die Möglichkeiten der Version 4.7 bzw. 5.x verwendet werden sollen.

Was bedeutet die Fehlermeldung "Wertekollision" und was ist zu tun?

In einer Datei gibt es mindestens zwei Datensätze, die für einen eindeutigen Schlüssel übereinstimmende Schlüsselwerte haben.

Dieser Zustand kann z. B. dadurch auftreten, dass in der Konfiguration bei einer nicht leeren Datei ein Schlüssel von "nicht eindeutig" auf "eindeutig" umgestellt wurde.

Als Abhilfe kann man den Schlüssel zunächst wieder auf "nicht eindeutig" zurückstellen und prozedural alle Datensätze über diesen Schlüssel lesen. Beim Aufeinanderfolgen von identischen Schlüsselwerten können die betreffenden Datensätze speziell behandelt werden, also entweder unter einem anderen Schlüsselwert gespeichert oder nach einer Sicherheitsabfrage gelöscht werden. Im ersteren Fall sollte vor der Speicherung der Schlüsselwert auf Eindeutigkeit getestet werden. Dies geschieht mit dem Befehl RecRead() unter Verwendung der Option \_RecNoLoad.

Anschließend kann der Schlüssel endgültig auf "eindeutig" umgestellt werden.

Nach jeder Veränderung von Schlüsseln, muss eine Schlüsselreorganisation durchgeführt werden.



Falls die Meldung erscheint, ohne dass ein Schlüssel umgestellt wurde, ist ein Defekt in der Datenbank zu vermuten. Hier sollten, wie bei Defekten üblich, die Errorlogdateien untersucht werden. Es sollte ein Backup der Datenbank und anschließend eine Erweiterte Diagnose mit Schlüsselanalyse durchgeführt werden. Welche weiteren Maßnahmen erforderlich sind, hängt vom Ergebnis der Diagnose bzw. automatischen Reparatur ab.

Wie kann eine automatische Versionsüberprüfung des CONZEPT 16-Clients beim Start des Systems erfolgen?

Mit dem folgenden Skript wird die Datei C16\_WINC.EXE in zwei unterschiedlichen Ordnern miteinander verglichen. Existieren die Zielordner oder die Datei im Zielordner nicht, wird das komplette Quell-Verzeichnis kopiert.



Wird das Skript beim Anmelden des Benutzers ausgeführt, kann eine Überprüfung des lokalen CONZEPT 16-Clients mit einem Client, der an einer zentralen Stelle im Netzwerk abgelegt ist, stattfinden. Besonders nützlich ist das Skript in Verbindung mit der Hot-Standby Option, wenn der primäre Datenbank-Server zugleich die Aufgaben des File-Servers übernimmt. Fällt der Primär-Server aus, können die Clients trotzdem weiterarbeiten, da sie auf den lokalen Festplatten abgelegt sind. Eine umfangreiche Administration bei einem Update des CONZEPT 16-Clients entfällt, da bei jedem Anmelden an das System der Versionsstand überprüft wird.

Kern des Skripts ist der Vergleich zweier Dateien. Der Vergleich findet durch den Befehl COMP statt. Das Ergebnis wird in eine externe Datei geschrieben. Der Befehl erwartet nach dem Vergleich eine Eingabe. Diese Eingabe steht in der Datei INPUT.TXT und besteht aus "n <CR>". Die Ausgabe von COMP wird mit FIND durchsucht. Wird der Text "Dateien sind identisch" nicht gefunden, erfolgt das Kopieren der Dateien aus dem Originalpfad in den lokalen Pfad.

### VersionCheck.CMD

```
@ECHO OFF IF "%1"=="/?" GOTO HELPIF not exist P:\ORG\c16\c16_winc.exe GOTO ENDIF not exist c:\c16
```

### INPUT.TXT

n

Welche Informationen können der CONZEPT 16-Lizenznummer entnommen werden? Die neuen Lizenzen werden in zwei Lizenzprodukte und vier Editionen unterschieden. Das Lizenzprodukt steht im ersten Zeichen und kann folgende Werte annehmen:

- **C** CONZEPT 16 - Dongle
- **T** TLL - Time-limited License

Dem zweiten Zeichen kann die Lizenzedition entnommen werden:

- **D** Developer-Edition
- **S** Standard-Edition
- **A** Advanced-Edition
- **E** Enterprise-Edition

Nach dem Lizenztyp wird eine sechstellige Zahl angegeben, die größer als 100.000 ist. Diese Lizenznummer ist eindeutig. Anschließend folgt das Lizenzsuffix.

### Das neunte Zeichen - Das Betriebssystem

Im ersten Zeichen wird das Betriebssystem der Lizenz angegeben:

- **W** Microsoft Windows-Betriebssysteme
- **L** Linux-Betriebssysteme
- **M** Multilizenz sowohl für Microsoft Windows XP, 2003, Vista, 2008, 7, 2008 R2, 8, 2012, als auch für Linux

### Das zehnte Zeichen - Der Lizenzschutz

Die Art des Lizenzschutzes wird in diesem Zeichen angegeben:

- **U** USB-Dongle
- **P** Parallel-Port-Dongle
- **N** Internet basierte Lizenz
- **H** Lizenz des Sekundär-Servers bei einem Hot-Standby-System. Dieser benötigt kein Dongle.

### Das elfte Zeichen - Optionen

Folgende Lizenzoptionen sind möglich:

- **H** Hot-Standby-Option
- **D** Demo-Version

Mit einer Lizenz mit der Hot-Standby-Option "/H" können zwei Server betrieben werden. Die Lizenz wird dazu auf dem Primärserver eingesetzt. Das Sekundärsystem besitzt nach der Inbetriebnahme von Hot-Standby die gleiche Lizenznummer ohne die Option, lediglich beim Zeichen für den Lizenzschutz wird ein "H" angegeben. Die Demo-Version "/D" kann bei potentiellen Neukunden installiert werden, um die Funktionsweise des Programms im Umfeld des Neukunden zu demonstrieren. Sie ist zeitlich limitiert. Bis auf die zeitliche Limitierung entspricht sie der gleichen Lizenz ohne die Option "/D".

Wie können die Releasestände von Windows ermittelt werden?

Unter Windows kann die Windowsversion über den Eintrag "Eigenschaften" des Kontextmenüs des Arbeitsplatzes oder über das Programm winver.exe ermittelt werden.

5.1 (2600)	Windows XP
5.1 (3790)	Windows Server 2003
6.0 (Build 6000)	Windows Vista
6.0 (Build 6001)	Windows Server 2008
6.1 (Build 7600)	Windows 7
6.1 (Build 7600)	Windows Server 2008 R2
6.2 (Build 9200)	Windows 8
6.2 (Build 9200)	Windows Server 2012
6.3 (Build 9600)	Windows 8.1
6.3 (Build 9600)	Windows Server 2012 R2
10.0 (Build 10240)	Windows 10
10.0 (Build 14393)	Windows Server 2016



Unter Windows XP befindet sich der Arbeitsplatz im Startmenü. Bei Windows Vista müssen die Eigenschaften des Computers aufgerufen werden.

Wie kann unter Windows-Betriebssystemen die Version des installierten Service Packs ermittelt werden?

Das Service Pack kann über das Programm winver.exe ermittelt werden.

## Kontakt

Alternativ kann diese Information auch über den Explorer über **Hilfe / Info** abgerufen werden.

### **Beispiel:**

Version 5.0 (Build 2195: Service Pack 3)

Glossar  
Begriffserklärungen

• **Benutzer**

Der Benutzer meldet sich in der Datenbank an. Zu einem Benutzer gehört in der Regel ein Kennwort, das bei der Anmeldung mit angegeben werden muss. In der Evaluierungsversion kann die Anmeldung mit dem Benutzer "Start" oder "Dev" erfolgen. Zur Anmeldung wird kein Kennwort benötigt.

• **CA1 ... CA8**

Datenräume einer CONZEPT 16-Datenbank. In den Datenräumen werden die Datenstruktur, Dialoge, Prozeduren und Datensätze gespeichert. Übersteigt die Datenmenge die maximale Dateigröße des Betriebssystems, auf dem der CONZEPT 16-Server installiert ist, wird automatisch ein weiterer Datenraum angelegt. Es können bis zu acht Datenräume angelegt werden.

Die Aufteilung der Datenbank in mehrere Datenräume ist mit aktuellen Betriebssystemen nicht mehr notwendig. Mehrere Datenräume können in diesem Fall verwendet werden, um eine Sicherung auf bestimmten Medien zu erleichtern.

• **Client**

Bestandteil der Architektur des Client/Server-Konzepts. Der Client stellt das Benutzerinterface zur Verfügung. Die Datenverarbeitung findet durch den Server statt. Durch die Aufgabentrennung wird ein Höchstmaß an Sicherheit und Performance erreicht.

• **Datei**

Die Datei ist ein Bestandteil der Datenstruktur. Sie entspricht einer Tabelle. In einer Datei werden alle Datensätze zu einem Thema (Kundenadressen, Artikel usw.) gespeichert. Innerhalb einer Datei werden unter anderem Teildatensätze<sup>?</sup>, Felder<sup>?</sup> und Schlüssel<sup>?</sup> angelegt.

• **Datenbank, relationale**

Ein Datenbankmodell, das auf Entitäten und deren Verhältnissen (Relationen) basiert. Jede Entität (zum Beispiel ein Artikel) verfügt über verschiedene Eigenschaften (Liefermenge, Preis usw.). Steht eine Entität zu einer anderen Entität in Beziehung, wird dies durch eine Relation dargestellt. So könnte zum Beispiel "wurde bestellt von" eine Relation zwischen Artikel und Kunde sein. Über diese Relationen können Abfragen an die Datenbank gestellt werden (zum Beispiel können alle Kunden ermittelt werden, die einen bestimmten Artikel bestellt haben).

• **Datensatzpuffer**

siehe Feldpuffer<sup>?</sup>


• **Datenstruktur**

Mit der Datenstruktur wird der Inhalt eines Datensatzes festgelegt. In der Datenstruktur wird festgelegt, in welcher Reihenfolge, welche Daten abgelegt werden. Darüber hinaus werden hier die Schlüssel (Indizes) festgelegt.

### • Dongle

Der Dongle schützt die Lizenz vor illegalem Kopieren. Innerhalb der Applikation kann die CONZEPT 16-Lizenznummer verwendet werden, um bestimmte Module der Applikation freizuschalten. Damit profitiert auch der Anwendungsentwickler vom Lizenzschutz. Die vectorsoft AG liefert zwei unterschiedliche Dongle aus: den Parallelport-Dongle und den USB-Dongle. Damit der Dongle angesprochen werden kann, muss ein Treiber installiert werden. Der Treiber steht ebenso wie die Programmstände im Kundenbereich der Homepage der vectorsoft AG zum Download bereit.

### • Ereignis

Ein Ereignis wird durch den Benutzer oder das Betriebssystem ausgelöst. Klickt zum Beispiel ein Benutzer eine Schaltfläche an, vergrößert ein Fenster, wählt in einer Liste einen Datensatz aus oder wird in einem überwachten Verzeichnis eine Datei angelegt, wird ein Ereignis ausgelöst. Ist bei diesem Ereignis eine Funktion  eingetragen, wird diese Funktion ausgeführt. So werden bestimmte Verarbeitungen der Applikation an Aktionen des Benutzers gebunden.

### • Evaluierungslizenz

Mit der Evaluierungslizenz kann die Entwicklungsumgebung von CONZEPT 16 ausprobiert werden. Die Evaluierungslizenz wird zusammen mit Beispieldatenbanken, einem Tutorial und einer leeren Datenbank für eigene Entwicklungen ausgeliefert. Neben dem Produkt steht auch der Support für einen Zeitraum von 30 Tagen zur Verfügung.


Weitere Informationen bekommen Sie gerne von unserem Vertrieb:

Tel.: +49 6102/660200  
E-Mail: [sales@vectorsoft.de](mailto:sales@vectorsoft.de)



### • Feld

Die Spalte einer Tabelle. Die einzelnen Informationen zu einem Datensatz werden in mehreren Feldern gespeichert. Der Datensatz eines Kunden besteht zum Beispiel aus den Feldern Name, Straße, Ort, laufender Umsatz usw. Jedem Feld werden Informationen eines Datentyps zugewiesen (Name ist vom Typ Alphanumerisch, laufender Umsatz vom Typ Fließkomma usw.).

### • Feldpuffer

Wird ein Datensatz gelesen, stehen die Inhalte des Datensatzes im Speicher zur Verfügung. Dieser Speicher ist der sogenannte Feldpuffer. Der Feldpuffer wird durch die Namen der Felder  angesprochen.

### • Funktionen

Funktionen werden innerhalb von Prozeduren  definiert. In diesen Funktionen findet die Verarbeitung der Applikation statt. Funktionen können von anderen Funktionen oder Ereignissen  aufgerufen werden. Eine Funktion wird entweder mit main oder sub definiert.

### • Index

siehe Schlüssel 

- **Key**

siehe Schlüssel<sup>?</sup>

- **Lizenz**

Eine CONZEPT 16-Lizenz erlaubt den Betrieb eines CONZEPT 16-Servers<sup>?</sup> mit einer bestimmten Anzahl von gleichzeitigen Benutzern auf unterschiedlichen Rechnern. Die maximale Anzahl der Benutzer ist nicht an die Anzahl der Benutzer im Netz gekoppelt. Alle anderen Bestandteile einer CONZEPT 16-Installation (Client, ODBC-Treiber usw.) müssen nicht lizenziert werden. Eine Lizenz besteht aus einem Dongle<sup>?</sup> und einer Lizenzdatei<sup>?</sup>. Weitere Informationen befinden sich im Abschnitt CONZEPT 16-Server - Lizenzmanagement.

- **Lizenzdatei**

Die Lizenzdatei (c16.lic) wird für den Betrieb des CONZEPT 16-Servers benötigt. Diese Datei enthält die Leistungsmerkmale der eingesetzten Lizenz. Die Lizenzdatei steht im Kundenbereich der Homepage der vectorsoft AG zum Download bereit. Änderungen an der Lizenz werden durch eine Aktualisierung der Lizenzdatei übertragen.

- **ODBC (Open DataBase Connectivity)**

Diese Schnittstelle steht unter Windows-Betriebssystemen zur Verfügung. Sie erlaubt den Zugriff von CONZEPT 16 auf beliebige Datenquellen. Ebenso kann eine CONZEPT 16-Datenbank als Datenquelle eingerichtet werden. Nähere Informationen befinden sich im Abschnitt Die ODBC-Schnittstelle.

- **Prozeduren**

Prozeduren bestehen aus einer oder mehreren Funktionen<sup>?</sup>. Innerhalb einer Prozedur sollten alle Funktionen, die zu einem Themenbereich (einem Programmmodul oder Applikationsschicht) gehören definiert werden.

- **Protokoll**

Die Kommunikation zwischen den CONZEPT 16-Clients und dem Server wird über ein Protokoll abgewickelt. Das Protokoll stellt Basisfunktionalitäten zur Kommunikation zur Verfügung. Der Client und der Server kann über das Protokoll und TCP/IP kommunizieren.

- **Schlüssel**

Über Schlüssel oder Indizes werden die Datensätze einer Datei sortiert. Wird nach einem bestimmten Schlüsselwert gesucht, kann der Datensatz, der in dem entsprechenden Schlüsselfeld über diesen Wert verfügt, sofort gelesen werden.

- **Selektion**

Eine Selektion ist eine Teilmenge der Datensätze, die in einer Datei<sup>?</sup> enthalten sind. Die Teilmenge wird durch Bedingungen erstellt. In der Selektionsmenge sind alle Datensätze enthalten, die die Selektionsbedingungen erfüllen.

- **Server**

siehe Client<sup>?</sup>

- **Storage-Objekte**

Storage-Objekte sind alle Applikationsressourcen, die in der Datenbank angelegt bzw. gespeichert sind. Zu diesen Ressourcen zählen Dialoge, Menüs, die Druck-Objekte und Bilder, die im Dialog mit dem Programmierer erstellt oder über die Entwicklungsumgebung eingelesen wurden.

- **Teildatensatz**

Der Teildatensatz gruppiert eine Menge von Feldern. Auf diese Weise kann eine Datei (Tabelle) weiter unterteilt werden. Teildatensätze können abhängig von Feldinhalten aktiviert und deaktiviert werden, sodass eine variable Datenstruktur realisiert werden kann.

- **User**

siehe Benutzer 

- **UTC (Universal Time Coordinated)**

Die UTC ist eine Kombination aus der internationalen Atomzeit und der Universalzeit. Sie stellt eine weltweit einheitliche Uhrzeit dar. Die Lokalzeit wird ausgehend von UTC berechnet. Die mitteleuropäische Zeit berechnet sich aus UTC + 1 bzw. im Sommer aus UTC + 2.

- **Verknüpfung**

Mit Hilfe von Verknüpfungen werden die Relationen innerhalb der Datenbank realisiert. Eine Verknüpfung hat immer eine Ausgangsdatei (oder Quelle) und eine Zieldatei. In der Ausgangsdatei wird ein Verknüpfungsfeld und in der Zieldatei ein Schlüssel bestimmt. Wird in der Ausgangsdatei ein Datensatz gelesen, gelten alle Datensätze in der Zieldatei mit diesem Datensatz als verknüpft, die den Wert des Verknüpfungsfeldes als Schlüsselwert besitzen.

Beispiel:

In der Kundendatei gibt es Feld "Kundennummer". In der Auftragsdatei gibt es das gleiche Feld. Hier ist auch ein Schlüssel über dieses Feld definiert. In der Kundendatei kann eine Verknüpfung definiert werden, sodass alle Aufträge zu einem Kunden verknüpft sind.