Dokumentation der COM-Schnittstelle SelectLine Rechnungswesen Version 11.0

Inhaltsverzeichnis

0. Änderungen

1. Rewe 11.0

- *Neu:* Funktionen zur Ermittlung von Mandanteneinstellungen und Konfigurationsoptionen: **XFibu.LiesOptionen**, **XFibu.GetMandField**.
- *Neu:* Funktion **XFibu.GetJahrByDate** zur Ermittlung der Jahres-Datenbank für ein Datum.
- *Neu:* **XFibu.EditData** zur Bearbeitung von Daten über modale Stammdatendialoge.
- *Neu:* **XFibu.StornoSatzOptionen** zur Stornierung von Buchungssätzen und Angabe von Optionen.
- *Änderung:* Das Verhalten der COM-Komponente **OleTable** wurde überarbeitet und hinsichtlich SQL optimiert. Dadurch hat sich auch das Verhalten von **OleTable.RecordCount** minimal geändert (s. Abschnitt „Änderungen an OleTable ab Rewe V. 11.0“).
- *Ergänzung der Dokumentation:* Der Abschnitt Tipps & Tricks ist neu und enthält u. a. das Thema "Über COM aufgerufene Dialoge erhalten keinen Fensterfokus und öffnen

sich im Hintergrund".

2. Rewe 10.0.4

- *Neu:* Mit der Funktion **XFibu.HatLizenz** können die freigeschalteten Lizenzen der registrierten COM-Rewe abgefragt werden.

3. Rewe 10.0.0

- *Neu:* Prozedur **XFibu.StartMakroEx** sowie das Interface **IMakroParamList** zum Ausführen von Makros mit Parameterübergabe.
- *Neu:* Funktion **XFibu.StornoSatzPeriode** zur Stornierung von Buchungssätzen mit optionaler Beachtung der Buchungsperiode.

1. Einleitung

Die COM-Schnittstelle bietet bis jetzt folgenden Funktionsumfang
Mit dem Tabellenobjekt können Datensätze gelesen, editiert und neue zugefügt werden.
Das Buchungsobjekt beherrscht Einzel- und Sammelbuchungen.
Die Daten der Mandanten und beliebige Stammdaten werden nun aus Listen gewählt.

De- und Installation des COM-Servers erfolgt über das einmalige Ausführen der Programmdatei mit folgenden Parametern.
(XXX = BDE/SQL - je nach verwendeter Datenbankanbindung)

1. Installation und Deinstallation des COM-Servers

Registrieren:

ReweXXX.exe /regserver

Deregistrieren:

ReweXXX.exe /unregserver

Hinweis:
Bitte achten Sie darauf in der (Systemsteuernung>) BDE-Konfiguration, die *Option* auf **Local Share = True** einzustellen.

2. Das Rewe-Objekt (XFibu - IApp)

Dieses Objekt beinhaltet Funktionen für das Anlegen von Tabellen- und Buchungsobjekten sowie die Daten und Mandantenauswahl. Es wird innerhalb des Automatisierungskontrollers mit dem Namen "XFibu.App" initialisiert.

Hinweis:
Auf Grund der Kompatibilität zu älteren Versionen wurde der Name "XFibu" des Rewe-Objekts beibehalten.

Beispiel für Visual Basic

```
Set Rewe = CreateObject("XFibu.App")
While Not Rewe.ApplicationReady
Wend
Rewe.Login("Nutzerkürzel", "Passwort")
```

Die folgenden Funktionen beziehen sich dann beim Aufruf auf dieses Objekt.

Beispiel für Delphi

```
Table := Rewe.CreateTable('KU');
```

Funktion	Art	Beschreibung
ApplicationReady	variant	Zeigt ob das Programm vollständig gestartet wurde.
Login (Name, Passwort: string)	boolean	Der Aufruf dieser Funktion ist erforderlich, wenn Benutzer angelegt sind und in der DATEN.INI folgender Eintrag steht. [SYSTEM] OLELOGIN=0 Mit OLELOGIN=1 oder löschen des Eintrags wird das Passwort bei Programmstart vom Programm abgefragt.
GetMandant (Mandant, Jahr, Monat: string; Auswahl: integer)	variant	Diese Funktion dient der Mandantenauswahl; Auswahl = 0: Der Mandant wird automatische gewechselt. Auswahl = 1: Der Auswahldialog wird angezeigt. Der Parameter Mandant gibt die Mandantennummer, der Paramater Jahr das Buchungsjahr und der Parameter Monat die Startperiode des Buchungsjahres an. Der Parameter Monat ist optional. Wird kein Monat angegeben sucht die Rewe den ersten Buchungszeitraum im angegebenen Jahr. Der Monat ist normalerweise immer 1, außer bei abweichendem Wirtschaftsjahr. Ergebnis: Die Nummer des aktiven Mandanten.
GetMandantByDate(Mandant: String; Datum: TDateTime)	string	Wählt den Mandant und Jahr anhand des Belegdatums, d. h. das Programm „steht" anschließend in der angegebenen Kombination Mandant/Jahr.
Mandant	string	Liefert den aktuellen Mandanten.
JahrMonat	string	Liefert das aktuelle Bearbeitungsjahr und den Startmonat in der Form JJJJMM als Zeichenkette.

NextBeleg	variant	Liefert die nächste Belegnummer.
CreateBuchung	IBuchung	Mit dieser Funktion wird ein Buchungsobjekt angelegt.
CreateSaldo	IOleSaldo	Mit dieser Funktion wird ein Saldoobjekt angelegt.
ReadOpos (OPNr: integer)	IOleOPOS	Funktion zum Einlesen eines OP-Objekts mit der OPNummer „OPNr".
ReadFibuOpos (SatzNr, Position: Integer)	IOleOPOS	Funktion zum Einlesen eines OP-Objekts mit der Buchunssatznummer „SatzNr" und der Position „Position".
SelectData (BlobKey: WideString; LastKey: WideString)	variant	Diese Funktion aktiviert einen Auswahldialog in Form einer Liste. Der Parameter Blobkey legt dabei fest, um welche Tabelle es sich dabei handelt. Er hat die gleiche Bedeutung wie bei der Funktion CreateTable. Der Parameter LastKey markiert den Datensatz mit dem angegebenen Schlüssel.
EditData (BlobKey, Key: WideString);	OleVariant	Bearbeitung eines Datensatzes über den modalen Stammdatendialog.
CreateTable (ABlobKey: WideString)	IOleTable	Diese Funktion initialisiert ein Tabellenobjekt. Der Parameter TableKey enthält einen zweistelligen String aus Grossbuchstaben um eine Tabelle auszuwählen. Das Handling ob es sich dabei um Tabellen aus dem Daten- oder Mandantenverzeichnis, wird dadurch gekapselt. Die Liste der Schlüssel befindet sich im Anhang.
StornoSatz (Satz: Integer; Klasse: string)	boolean	Storniert die Buchung mit der Nummer SatzNr. Mit dem Parameter Klasse wird die Programmklasse angegeben (z.B.: F=Wawi, B=Rewe, K=Kasse). Im Fehlerfall wird eine Exception ausgelöst.
StornoSatzPeriode (Satz: Integer; Klasse: string; PeriodenAbschlussBeachten: boolean)	boolean	Storniert die Buchung mit der Nummer SatzNr. Mit dem Parameter Klasse wird die Programmklasse angegeben (z.B.: F=Wawi, B=Rewe, K=Kasse). Der Parameter PeriodenAbschlussBeachten steuert, ob der Periodenabschluss im Rechnungswesen beachten werden soll. Im Fehlerfall wird eine Exception ausgelöst.
StornoSatzOptionen (var Optionen: WideString)	Integer	Stornierung einer Buchung mit Angabe von Optionen. Mit dieser Funktion kann eine Buchung unter Angabe von Optionen storniert werden. Im Gegensatz zu den Funktionen XFibu.StornoSatz und XFibu.StornoSatzPeriode ist diese Funktion eine "echte" COM-Funktion ohne Anzeige von Dialogfenstern. Während bei den anderen beiden Funktionen unter Umständen Dialogfenster erscheinen (z. B. bei falscher Periode), kehrt diese Funktion mit Fehler zurück. Mit der Option "Exceptions" kann das Verhalten im Fehlerfall vorgegeben werden.

Wie bei der Funktion XFibu.LiesOptionen werden Optionen durch Optionsname;Wert angegeben, wobei einzelne Optionen durch Zeilenumbruch (CRLF, $13$10) zu trennen sind. Groß- und Kleinschreibung der Optionsnamen wird nicht unterschieden.
Zur Zeit (Stand Rewe 11.0) umfassen die Optionen die Möglichkeiten der Funktion XFibu.StornoSatzPeriode sowie zusätzlich die Festlegung, ob im Fehlerfall Exceptions generiert werden sollen oder nicht.
Folgende Optionen sind möglich:

- Exceptions
  Bestimmt, ob diese Funktion bei einem Fehler einen Rückgabewert < 0 zurückgibt oder eine OleException generiert wird:
  Exceptions;0 - keine Exceptions, Rückgabewert < 0
  Exceptions;1 - Exception im Fehlerfall
  Vorgabe (keine Angabe in Optionen) ist "Exceptions;0"
  Diese Option sollte als erster Parameter in der String-Liste angegeben werden, da bereits beim Auswerten der String-Liste Fehler auftreten können.
- SatzNr Angabe der Buchungssatznummer.
  Beispiel: SatzNr;1234
- OPQuelle OP-Quelle: B - Rewe, F - Wawi, K - Kasse
- PeriodenAbschluss Periodenabschluß beachten: 0 = nein, 1 = ja
  Beispiel: PeriodenAbschluss;1

Rückgabewerte (in Abhängigkeit von der Option "Exceptions"):
> 0 = SatzNr. (Erfolg)
< 0 = Fehler oder Exception

Fehlercode	OleException-Fehlercode	Beschreibung
> 0	---	kein Fehler Rückgabe: SatzNr.
- 5	$8004 1005	ungültige Option
- 6	$8004 1006	ungültige SatzNr.
- 7	$8004 1007	ungültige OP-Quelle
- 8	$8004 1008	fehlende Rechte für Funktion
- 9	$8004 1009	Buchungsjahr ist bereits abgeschlossen
- 10	$8004 100A	Buchungssatz

		nicht gefunden
- 11	$8004 100B	allg. Storno-Fehler (alle nicht weiter spezifizierten Fehler)

CreateQuery (const ADataBaseName: WideString)	IOleQuery	Diese Funktion initialisiert ein Queryobjekt. Der Parameter ADataBaseName enthält optional den logischen Verzeichnisnamen DATEN für Datenverzeichnis MANDANT für Mandantenverzeichnis.
MitKostenrechnung	boolean	Gibt zurück, ob die Kostenrechnung beim aktuell gewählten Mandanten aktiv ist.
LoginReady	variant	Zeigt ob das Login vollständig ausgeführt wurde. *(BDE/ADS immer TRUE, SQL TRUE wenn Login erfolgreich)*
StartMakro (const MakroDatei: WideString)		Ausführen eines Makros.
StartMakroEx (const MakroDatei: WideString, const ParamList: MakroParamList)		Ausführen eines Makros mit Parametern.
CreateMakroParamList	IMakro ParamList	Erstellen einer Liste von Parametern.
HatLizenz(aLizenzOption: LizenzOption)	WideString	Abfrage einer Lizenzausprägung. Wenn die Lizenz vorhanden ist, wird „J" zurückgegeben, sonst „N". Folgende Lizenzen können abgefragt werden: 0 = lizenzDemo 1 = lizenzStandard 2 = lizenzStandardPlus 3 = lizenzGold 4 = lizenzPlatin 5 = lizenzDiamond 6 = lizenzMaskenEditor 7 = lizenzToolboxRuntime 8 = lizenzToolboxEdit 9 = lizenzCobraErpProvider 10 = lizenzSprache 11 = lizenzEasy 12 = lizenzStampit 13 = lizenzWawiStandort 14 = lizenzWawiIntra 15 = lizenzWawiMosaic 16 = lizenzWawiMapkit 17 = lizenzWawiXtrade 18 = lizenzReweFibu 19 = lizenzReweKonas 20 = lizenzReweKonsul 21 = lizenzReweChOPOS 22 = lizenzReweAnlag

		23 = lizenzReweKost 24 = lizenzLohnKUG 25 = lizenzLohnFlexi 26 = lizenzLohnZVK 27 = lizenzLohnPfaendung 28 = lizenzLohnDakota 29 = lizenzLohnBau 30 = lizenzVertriebsModul
LiesOptionen (var Optionen: WideString)	WordBool	Auslesen von Mandanteneinstellungen und Konfigurationsoptionen. Diese Funktion liest aktuell gesetzte Optionen aus, wobei mehrere Optionen mit einem Aufruf ausgelesen werden können. Der Parameter Optionen ist eine aus einzelnen Zeilen bestehende String-Liste (pro Parameter eine Zeile). Die einzelnen Zeilen sind durch Wagenrücklauf/Zeilenvorschub (CR/LF, $13/$10) zu trennen. Groß- und Kleinschreibung wird bei den Optionsnamen nicht unterschieden. Leerzeilen im Parameter Optionen werden ignoriert. Für die Rückgabe der Werte wird an jeden Parameter - falls noch nicht vorhanden - ein Semikolon (";") angehängt. Auf das Semikolon folgt der aktuelle Wert. Boole'sche Optionen werden mit Wert "0" (false) oder "1" (true) zurückgegeben. Beachtet werden muß, daß - abhängig von den abzufragenden Optionen - bei der Rückgabe der Parameter Optionen mehr Zeilen enthalten kann, als beim Aufruf! Eine direkte Abfrage eines Parameters über den Listenindex ist nur möglich, wenn "einzeilige" Optionen abgefragt werden. Der eigentliche Rückgabewert der Funktion ist false (=0) bei einem Fehler, sonst true (<>0). Achtung! Der Parameter Optionen wird in jedem Fall geändert: Entweder werden die aktuellen Einstellungen zurückgegeben (Format: Option;Wert) oder es wird im Fehlerfall (->ungültige Option - Exception) die erste ungültige Zeile mit einem Fehlertext beschrieben. Beschreibung der Parametersyntax: s. Parameterformate für XFibu.LiesOptionen
GetMandField (const Feldname: WideString);	OleVariant	Mit dieser Funktion können Eigenschaften des internen Mandantenobjektes ausgelesen werden. Übergeben wird der Eigenschaftsname, wobei Groß-/ Kleinschreibung nicht unterschieden wird. Der Rückgabewert ist abhängig vom Typ der Eigenschaft.

		Die selbe Funktionalität steht in der Funktion XFibu.LiesOptionen bei Verwendung des Parameterformats "Mandant.Option..." zur Verfügung. Achtung! Es wird nicht garantiert, daß Eigenschaften in späteren Versionen noch zur Verfügung stehen. Beispiele: • SteuerIdentNr - String • BuchPerioden - Integer • EuroEinfuehren - Boolean
function GetJahrByDate (const Datum: TDateTime);	WideString	Ermittlung der Jahres-Datenbank zu einem Datum. Mit dieser Funktion wird zum übergebenen Datum die zugehörige Jahres-Datenbank des aktuellen Mandanten ermittelt. Dabei spielt es keine Rolle, in welchem Jahr der aktuelle Mandant gerade steht. Im Gegensatz zur Funktion XFibu.GetMandantByDate wechselt diese Funktion **nicht** in die Jahres-Datenbank des Parameters "Datum". Falls ein Wechsel notwendig ist, muß dies mit der Funktion XFibu.GetMandantByDate erfolgen. Bei ungültigen Datumsangaben (Datum liegt vor Start der ersten Jahres-DB bzw. nach Ende der letzten/aktuellen Jahres-Datenbank) wird ein Leerstring zurückgegeben. Der Rückgabewert umfaßt nicht den kompletten Namen der Jahres-Datenbank sondern nur Jahr und Monat im Format JJJJMM. Der echte Datenbankname kann dann zusammengesetzt werden und hat zur Zeit folgendes Format [eckige Klammern = Platzhalter]: SL_M[Mandantenname]-[JahresDB-Name] Beachtet werden muß, daß diese Funktion "relativ" viel Zeit benötigt. Pro Datum sollte deshalb nur ein Aufruf erfolgen. Beispiel: Mandant: WUNDF, Rückgabewert: 200907 -->vollständiger Jahres-Datenbankname: SL_MWUNDF-200907

3. Das Tabellen-Objekt (IOleTable)

Die Erzeugung des Objekts erfolgt mit der Funktion „CreateTable" des Rewe-Objektes.

Funktion	Art	Beschreibung
First		Der Datensatzzeiger stellt sich auf die erste Position.
Last		Der Datensatzzeiger stellt sich auf die letzte Position.
Next		Der Datensatzzeiger stellt sich auf die nächste Position.
Prior		Der Datensatzzeiger stellt sich auf die vorherige Position.
EoF	boolean	Diese Funktion liefert True, wenn der letzte Datensatz mit Next verlassen wurde oder kein Datensatz in der Tabelle existiert.
BoF	boolean	Diese Funktion liefert True, wenn der erste Datensatz mit Prior verlassen wurde oder kein Datensatz in der Tabelle existiert.
RecordCount	variant	Diese Funktion liefert die Anzahl der Datensätze der Tabelle.
GetValue (FieldName: String; Value: Variant)	variant	Der Inhalt des Feldes mit der Bezeichung *FieldName* wird zurückgeliefert.
PutValue (FieldName:String; Value: Variant)	variant	Der Inhalt des Feldes mit der Bezeichung *FieldName* wird mit dem Wert *Value* belegt. **Hinweis:** Die endgültige Speicherung von geänderten Werten erfolgt erst nach dem Aufruf der Funktion *Post*.
Append	variant	Ein neuer Datensatz wird an die Tabelle angehängt und steht zur Bearbeitung bereit.
Insert	boolean	Ein neuer Datensatz wird in die Tabelle eingefügt und steht zur Bearbeitung bereit. **Hinweis:** Bei Tabellen, die einen Primärindex besitzen (zusätzlich gibt es eine Datei mit der Endung PX), hat die Funktion *Append* die gleiche Wirkung wie die Funktion *Insert*.
Delete		Der aktuelle Datensatz wird gelöscht.
FindKey (Value: Variant)	variant	Der Wert *Value* wird über den Primärindex der Tabelle gesucht. Wird er gefunden, wird der Datensatzeiger entsprechend eingestellt und die Funktion liefert True.
FindKey2 (Value1, Value2: Variant)	variant	Wie „FindKey", sucht nach einem Datensatz mit dem zusammengesetzten Primärindex (Value1, Value2).
FindKey3 (Value1, Value2, Value3: Variant)	variant	Wie „FindKey", sucht nach einem Datensatz mit dem zusammengesetzten Primärindex (Value1, Value2, Value3).

Post	boolean	Der letzte Datenbank-Befehl wird explizit ausführt.
PutField (FieldName: String; Value: Variant)	variant	Belegt den Inhalt des Feldes mit der Bezeichung *FieldName* wird mit dem Wert *Value*. **Hinweis:** Die endgültige Speicherung von geänderten Werten erfolgt erst nach dem Aufruf der Funktion *Post*.
GetField (FieldName: String)	variant	Liefert den Inhalt des Feldes mit der Bezeichung *FieldName*.
GetFields	variant	Liefert Feld mit Namen, Datentyp und Größe der Tabellenfelder. Array[3*i+ 1] = Name Array[3*i+ 2] = Datentyp Array[3*i+ 3] = Feldlänge **Hinweis:** i = 0, 1... Anzahl der Tabellenfelder reduziert um 1
GetIndex	string	Liefert den aktuell gesetzten Index.
SetIndex (anIndex : String)	string	Setzt einen Index.
LastKey	string	Setzt den Datensatzeiger auf die zuletzt verwendete Tabelle mit dem entsprechenden Primärschlüssel.
Refresh		Aktualisiert die zwischengespeicherte Datenmenge.

Hinweis:
Die Rechte aus der Passwortverwaltung der Rewe hinsichtlich der Datenzugriffe (sehen, ändern, anlegen, löschen), werden auch durch den Zugriff über die OLE-Schnittstelle überprüft. Ggf. werden entsprechende Hinweise angezeigt und der Zugriff unterbunden.

1. Änderungen an OleTable ab Rewe V. 11.0

Mit der Version 11 von Warenwirtschaft und Rechnungswesen wurde das Verhalten der COM-Komponente OleTable überarbeitet und hinsichtlich SQL optimiert.
Bis einschließlich Version 10.1.x wurde beim Anfordern von Tabellen mit OleTable.CreateTable() die Datenbanktabelle komplett geöffnet und alle Datensätze mit „select *" sofort geholt. Selbst wenn später mit FindKey() nur bestimmte Datensätze gesucht wurden. Das Abfordern eigentlich überflüssiger Datensätze dauerte bei Tabellen mit vielen Datensätzen dementsprechend lange, teilweise bis in den Sekundenbereich hinein.
Ab der Version 11 wurde das Handling intern auf SQL-Zugriffe optimiert. Beim Anfordern eines OleTable-Objektes werden jetzt keine Datensätze mehr automatisch geholt. Erst bei weiteren Zugriffen werden nur die erforderlichen Daten abgerufen. Durch das neue Verhalten ergeben sich insbesondere bei großen Tabellen erhebliche Performancevorteile.
Als Folge dieser Optimierungen hat sich das Verhalten von OleTable.RecordCount() minimal verändert.

Änderungen an OleTable.RecordCount()
Bis Version 10.1.x war der Wert über die komplette „Lebenszeit" der OleTable konstant (abzüglich/zuzüglich gelöschter/eingefügter Datensätze über die OleTable). Änderungen außerhalb der verwendeten OleTable (z. B. durch andere Benutzer) waren grundsätzlich nicht sichtbar.
Ab Version 11 wird OleTable.RecordCount immer aktuell mit „select count(*)" geholt. Der Wert ist damit nicht mehr konstant, da Änderungen von außerhalb der OleTable in die Zählung mit

eingehen.
Durch diese Änderung hat RecordCount nur noch "informativen Charakter". Um alle Datensätze zu durchlaufen, sollten feste Schleifen über RecordCount vermieden werden. Stattdessen sollten unbestimmte Schleifen (repeat/while) bis OleTable.EoF durchlaufen werden.

Programmtechnische Überlegungen zur Verwendung von OleTable-Objekten:
Wird über COM auf SelectLine SQL-Programme mit Versionsnummer 10.x und kleiner zugegriffen, so kann es bei Verwendung von OleTable-Objekten auf Tabellen mit vielen Datensätzen zu Performance-Einbrüchen kommen. (Die BDE-Versionen sind nicht betroffen.) Hier ist die einzige Lösung, das COM-Programm auf OleQuery umzustellen. (Was wiederum bei BDE-Versionen langsam sein kann.)
Bei SelectLine SQL-Programmen ab Version 11 bestehen performance-technisch praktisch keine Unterschiede zwischen OleTable und OleQuery. Dies bedeutet, daß COM-Clients durch Umstellung des Servers auf Version 11 schneller laufen, wenn OleTable-Objekte verwendet werden.

Die folgende Tabelle gibt einen Überblick über die verschiedenen Kombinationsmöglichkeiten:

SelectLine COM-Server	OleTable	OleQuery
BDE V. <= 10.x	schnell	schnell nur langsam, wenn auf nicht-indizierte Felder zugegriffen wird
SQL V. <= 10.x	langsam bei vielen Datensätzen (->wg. select *) evtl. umstellen auf OleQuery, wenn COM-Client nie mit BDE-Versionen benutzt wird (da OleQuery dort evtl. langsam ist)	schnell Performance ist nur abhängig vom SQL Server
SQL V. >= 11.x	schnell aufgrund interner Optimierungen in OleTable COM-Clients, die OleTables verwenden, werden evtl. automatisch schneller	schnell Performance ist nur abhängig vom SQL Server

4. Das Query-Objekt (IOleQuery)

Die Erzeugung des Objekts erfolgt mit der Funktion „CreateQuery" des Rewe-Objektes.

Funktion	Art	Beschreibung
SetSQLText (SQLText: string)		SQL-Statement setzen.
OpenSQL		SQL-Statement anzeigen. (z.B. select-Anweisung)
ExecuteSQL		SQL-Statement ausführen. (z.B. update-, delete-, insert-Anweisungen)
CloseSQL		Datensatzzeiger schließen.
First		Der Datensatzzeiger stellt sich auf die erste Position.
Last		Der Datensatzzeiger stellt sich auf die letzte Position.
Next		Der Datensatzzeiger stellt sich auf die nächste Position.
Prior		Der Datensatzzeiger stellt sich auf die vorherige Position.
EoF	boolean	Diese Funktion liefert True, wenn der letzte Datensatz mit Next verlassen wurde oder kein Datensatz in der Tabelle existiert.
BoF	boolean	Diese Funktion liefert True, wenn der erste Datensatz mit Prior verlassen wurde oder kein Datensatz in der Tabelle existiert.
RecordCount	long	Diese Funktion liefert die Anzahl der Datensätze der Tabelle.
GetValue (FieldName: String; Var Value: Variant)	variant	Der Inhalt des Feldes mit der Bezeichung AFieldName wird zurückgeliefert.
GetField (FieldName: String)	variant	Der Inhalt des Feldes mit der Bezeichung AFieldName wird zurückgeliefert.

5. Das Buchungs-Objekt (IBuchung)

Mit dem Buchungs-Objekt können Einzel- und Sammelbuchungen erstellt werden. Das Objekt wird mit der Funktion „CreateBuchung" des Rewe-Objektes erstellt.

1. Einzelbuchung

EBBuchen (Datum, BelegNr, Konto, GegenKonto, Ust, Kst, FWCode, Text1, Text2, Betrag, FWBetrag, Skonto :Variant) : Variant

EBBuchenEx (Datum, BelegNr, Konto, GegenKonto, Ust, Kst, FWCode, Text1, Text2, Betrag, FWBetrag, Skonto, KLNr :Variant) : Variant

EBBuchenPeriode(Periode, Datum, BelegNr, Konto, GegenKonto, Ust, Kst, FWCode, Text1, Text2, Betrag, FWBetrag, Skonto, KLNr: Variant): Variant

Mit diesen Methoden werden Einzelbelege verbucht.
Im Fehlerfall geben die Funktionen einen Fehlertext, ansonsten einen String mit der Satznummer und der OPNummer durch ein Pipe-Zeichen „|" getrennt zurück. Fehlermeldungen enthalten kein Pipe-Zeichen.

Folgende Parameter werden übergeben

Datum	Datum der Buchung
BelegNr	Belegnummer
Konto	Konto der Buchung
GegenKonto	Gegenkonto der Buchung
Ust	Umsatzsteuerschlüssel. Wenn als Schlüssel 'Auto' übergeben wird, werden automatisch die bei den Konten hinterlegten Schlüssel genutzt.
Kst	Nummer der Kostenstelle
FWCode	Fremdwährungscode
Text1/2	Buchungstexte
Betrag	Buchungsbetrag
FWBetrag	Fremdwährungsbetrag laut FWCode Bei Fremdwährungsbuchuchungen müssen *Betrag* und *FWBetrag* übergeben werden. Die Umrechnung kann mit der Funktion *GetTagesWFaktorVonFW* des Rewe-Objektes erfolgen.
Skonto	Skontobetrag (Nur bei Buchungen mit Finanzkonto)
KLNr	Kunden bzw. Lieferantennummer (nur notwendig bei vom Personenkonto abweichender Kunden bzw. Lieferantennummer)
Periode	Buchungsperiode, in die die Buchung gebucht werden soll (Format YYYYMMX mit YYYY=Buchungsjahr, MM=Buchungsmonat, X=0 normaler Monat X=1..9 Abschlusspeiorde 1..9)

1. Sammelbuchung

SBKopf (Datum, BelegNr: Variant) : Variant

SBKopfPeriode(Periode: OleVariant; Datum: OleVariant; BelegNr: Variant) : Variant

Mit diesen Funktionen wird eine Sammelbuchung eröffnet. Ihr werden Datum, Belegnummer und bei der Funktion SBKopfPeriode die Periode, in die gebucht werden soll, übergeben. Die Periodenangabe muss dabei dem Format genügen, welches auch bei der Funktion EBBuchenPeriode verwendet wird (siehe vorherige Seite). Im Fehlerfall wird ein Fehlertext zurückgegeben, sonst leer.

SBAdd (KontoNr, Text1, Text2, FWCode, Ust, Kst, Betrag, FWBetrag, IstHaben: Variant) : Variant

Mit SBAdd wird eine weitere Position an eine Sammelbuchung angehängt. Im Fehlerfall wird ein Fehlertext zurückgegeben, sonst leer.

Es werden folgende Parameter übergeben

KontoNr	Konto der Buchung
Text1/2	Buchungstexte
FWCode	Fremdwährungscode
Ust	Umsatzsteuerschlüssel Wenn als Schlüssel 'Auto' übergeben wird, wird der bei den Konten hinterlegte Schlüssel genutzt.
Kst	Nummer der Kostenstelle
Betrag	Buchungsbetrag
FWBetrag	Betrag in der Fremdwährung laut FWCode
IstHaben	Boolean. Ist True, wenn der Buchungsbetrag auf der Habenseite ist.

SBAddEx (KontoNr, KLNr, Text1, Text2, FWCode, Ust, Kst, Ktr, Betrag, FWBetrag, IstHaben: Variant): Variant

Mit SBAddEx wird eine weitere Position an eine Sammelbuchung angehängt. Im Fehlerfall wird ein Fehlertext zurückgegeben, sonst leer.

Es werden folgende Parameter übergeben

KontoNr	Konto der Buchung
KLNr	Kunden- bzw. Lieferantennummer
Text1/2	Buchungstexte
FWCode	Fremdwährungscode
Ust	Umsatzsteuerschlüssel; wenn als Schlüssel 'Auto' übergeben wird, wird der bei den Konten hinterlegte Schlüssel genutzt.
Kst	Nummer der Kostenstelle
Ktr	Nummer des Kostenträgers
Betrag	Buchungsbetrag
FWBetrag	Betrag in der Fremdwährung laut FWCode
IstHaben	Boolean. Ist True, wenn der Buchungsbetrag auf der Habenseite ist.

SBBuchen: Variant;

Im Fehlerfall gibt die Funktion einen Fehlertext, ansonsten einen String mit der Satznummer und der OPNummer durch ein Pipe-Zeichen „|" getrennt zurück. Fehlermeldungen enthalten kein Pipe-Zeichen.

2. Weitere Funktionen

Eigenschaften	Zugriff	Art	Beschreibung
Sammel	in/out	boolean	Gibt an, ob es sich bei der Buchung um eine Sammelbuchung handelt.
Datum	in/out	Date	Gibt das Buchungsdatum der Buchung zurück. Das Datum kann auch verändert werden.
Beleg	in/out	string	Gibt den Buchbeleg der Buchung zurück. Der Beleg kann auch verändert werden.
Quelle	in/out	string	Einstelliges Kürzel für das Programm, welches bucht. Wird hier ein Wert <> 'B' angegeben, kann diese Buchung in der Finanzbuchhaltung nicht mehr geändert werden.
PeriodenAbschlussBeachten	in/out	boolean	Gibt an, ob bei der Buchung der Abschluss von Perioden eingehalten wird. Wird die Eigenschaft auf True gesetzt, werden Buchungen, die in abgeschlossene Perioden gebucht werden würden mit einer Fehlermeldung abgelehnt. Ist die Eigenschaft False, werden solche Buchungen in die nächste offene Buchungsperiode verschoben.

Hinweis:
Der Zugriff auf die Eigenschaften ist richtungsweisend zu verstehen. Für den lesenden Zugriff ist es „in", für den Schreibenden „out".

Funktion	Art	Beschreibung
GetBuchPos (Index: long)	IBuchPos	Liefert ein Positionsobjekt anhand seiner Position. 0 liefert die erste Position, 1 die zweite Position usw.
NewBuchPos	IBuchPos	Erstellt ein neues Positionsobjekt.

6. Das Positions-Objekt (IBuchPos)

Mit dem Positionsobjekt können Buchungspositionen erstellt werden. Das Objekt wird mit der Funktion „NewBuchPos" des Buchungsobjektes angelegt.

Eigenschaften	Zugriff	Art	Beschreibung
KontoSoll	in/out	string	das Soll-Konto
KontoHaben	in/out	string	das Haben-Konto
Waehrung	in/out	string	die Währung der Buchung
SteuerCode	in/out	string	den Steuer-Code der Buchung
Text1	in/out	string	den ersten Buchungstext
Text2	in/out	string	den zweiten Buchungstext
Brutto	in/out	double	den Brutto-Betrag
Netto	in/out	double	den Netto-Betrag
Skonto	in/out	double	den Skonto-Betrag
OPBeleg	in/out	double	den OP-Beleg
SteuerProzent	in/out	double	den Prozentsatz der Steuer
ZahlPlan	in	string	der ZahlPlan bzw. Ratenzahlung, wobei die Teilbeträge als Wertepaare Datum und Betrag übergeben werden. Diese sind durch ein Pipe-Zeichen „|" zu trennen.

Hinweis:
Der Zugriff auf die Eigenschaften ist Richtungsweisend zu verstehen. Für den lesenden Zugriff ist es „in", für den Schreibenden „out".

Funktion	Art	Beschreibung
Check (Check: long)	boolean	Prüft das Positionsobjekt. Der Parameter Check wird nicht verwendet. Im Fehlerfall wird eine Exception ausgelöst.
BucheWaehrung (nWaehrung: String; Kurstyp: long; LWKurs, FWKurs, FWBetrag, LWBetrag, EUBetrag: Double)	boolean	Nur bei Fremdwährungsbuchungen zu verwenden.
BucheUStKSt (UstCode, KSt1, KSt2, OPBeleg: String)	boolean	Übergabe von UstCode, OPBeleg und Kostenstellen.
BuchAdd (KontoNr, Text1, Text2, USt, KSt: String; Betrag: Double; IstHaben: Boolean)	boolean	Übergibt alle Werte einer Sammelbuchungsposition in enem Block.
GetOleOpos	IOleOPOS	Liefert das OPOS-Objekt zur Buchungsposition.
ChangeSteuerProzent (Prozent: Double)	boolean	Änderung der Steuerprozente, wenn diese vom Standard aus den Steuerschlüsseldaten abweichen.

7. Das Saldo-Objekt (ISaldo)

Mit dem Saldo-Objekt kann der Saldo eines Kontos abgefragt werden.

Funktion	Art	Beschreibung
Saldo (Konto : String)	double	Liefert der aktuellen Saldo eines Kontos.

8. Das OPOS-Objekt (IOleOPOS)

Mit diesem Objekt können die Daten zu einem „Offenen Posten" geändert werden. Das Objekt verfügt über folgende Eigenschaften und Methoden.

Eigenschaften	Zugriff	Art	Beschreibung
Bankverb	in/out	long	Bankverbindung des Debitors/Kreditors im OP
KLNr	in/out	string	Kunden- bzw. Lieferantennummer
Kassebank	in/out	string	Eigene Bankverbindung
Skonto1	in/out	double	Skonto1 in Prozent
Skonto2	in/out	double	Skonto2 in Prozent
SkontoTage1	in/out	long	Tage für Skonto 1
SkontoTage2	in/out	long	Tage für Skonto 2
Belegnummer	in/out	string	Das Feld OPBeleg
Refnummer	in/out	string	Das Feld Referenznummer
ZahlungszielTage	in/out	long	Tage bis zur OP-Fälligkeit
ZahlungsZiel	in/out	string	Setzt die Zahlungsbedingungen eines OP's anhand der Stammdaten
Provision	in/out	double	Das Feld Provision
Vertreter	in/out	string	Das Feld Vertreter
Skontofaehig	in/out	double	Skontofaehiger Betrag
Quelle	in/out	string	Das Feld Quelle; Hier können nutzerspezifische Informatiinen abgelegt werden.
Kostenstelle	in/out	string	Das Feld Kostenstelle
Kostentraeger	in/out	string	Das Feld Kostenträger

Hinweis:
Der Zugriff auf die Eigenschaften ist Richtungsweisend zu verstehen. Für den lesenden Zugriff ist es „in", für den Schreibenden „out".

Funktion	Art	Beschreibung
Save	string	Schreibt die geänderten Daten in die Datenbank.

9. Das MakroParamList-Objekt (IMakroParamList)

Mit diesem Objekt können Parameter für die Ausführung eines Makros in einer Liste hinterlegt werden.

Eigenschaften	Zugriff	Art	Beschreibung
Count	in/out	long	Anzahl der Parameter
Name [Index]	in/out	string	Liefert den Namen des Paramters
Value [Index]	in/out	string	Liefert den Wert des Parameters

Hinweis:
Der Zugriff auf die Eigenschaften ist Richtungsweisend zu verstehen. Für den lesenden Zugriff ist es „in", für den Schreibenden „out".

Funktion	Art	Beschreibung
AddParam	string	Schreibt die geänderten Daten in die Datenbank.

10. Tipps & Tricks, bekannte Probleme

Dieser Abschnitt enthält Hinweise zu möglichen Problemen und ihren Lösungsmöglichkeiten.

1. Die Funktion XFibu.ApplicationReady kehrt nie zurück, das COM-Programm scheint zu hängen

Bei einer hohen Belastung des Servers, im konkreten Fall liefen auf dem Server lokale Zugriffe und mehrere Terminalserver-Sessions auf der BDE-Rewe, kehrte das COM-Programm beim Warten auf den Start der Rewe aus der Schleife

while not Fibu.ApplicationReady do;

nie zurück. Der Grund: Zusätzlich zur hohen Serverbelastung während der Rewe-Initialisierung läuft das COM-Programm in einer Endlosschleife und wartet auf den Rewe-Start, was wiederum die Serverlast unnötigerweise erhöht. Hier empfiehlt es sich, das COM-Programm für eine gewisse Zeit stillzulegen:

```
while not Fibu.ApplicationReady do
    Sleep(100);          { Angabe in Millisekunden,
                           hier: eine Zehntelsekunde warten }
```

Der Delphi "Sleep"-Befehl wird direkt an den Microsoft Windows API-Befehl "Sleep" durchgereicht. Dieser bewirkt, daß der Thread seine Zeitscheibe nicht voll ausnutzt, sondern sofort unterbrochen wird, so daß der nächste Thread an die Reihe kommt.

Weitere Informationen finden sich in der MSDN Library in der Beschreibung der Funktionen "Sleep" bzw. "SleepEx". In anderen Programmiersprachen muß entsprechend nach Funktionen gesucht werden, die diese API-Aufrufe durchführen.

Es ist eine gute Idee, diesen Befehl generell einzubauen. Im Idealfall, wenn die sofort startet, "verschwendet" man einmalig je nach Parameter ein bißchen Zeit. In den hier in der Hilfe aufgeführten Code-Beispielen wurde der Befehl größtenteils weggelassen, um die Beispiele so einfach wie möglich zu halten.

2. Über COM aufgerufene Dialoge erhalten keinen Fensterfokus und öffnen sich im Hintergrund

Zur Erklärung dieses "Phänomens" ist etwas "Windows-Geschichte" notwendig:

Bis einschließlich Windows 98 bzw. - auf der ehemaligen "Profi-Schiene" - Windows NT wurden gestartete Programme immer im Vordergrund geöffnet. Das hatte z. B. zur Folge, daß während eines Programmstarts bediente andere Programme (oder das geöffnete Startmenü) plötzlich ihren Fokus verloren und sich das neue Programm nach vorn drängelte.
Ab Windows ME bzw. Windows 2000 hat Microsoft dieses Verhalten schrittweise geändert, so daß in aktuellen Windows-Versionen Programme im Hintergrund gestartet werden. Derart gestartete Programme machen sich seitdem mit einem Blinken in der Taskleiste bemerkbar.
Für die Interoperabilität verteilter Anwendungen mußte jetzt allerdings eine Möglichkeit geschaffen werden, um Dialogfenster gezielt im Vordergrund öffnen zu können:
Der (aufgerufene) Server (->SelectLine Wawi) muß dafür sorgen, daß neue Dialogfenster programmweit im Vordergrund geöffnet werden. Üblicherweise passiert das beim Öffnen neuer Dialoge durch die verwendeten Komponenten oder durch Aufruf der Windows-API-Funktion "BringToFront" oder ähnlicher Funktionen.
Der COM-Client wiederum muß dem Betriebssystem mitteilen, daß er anderen Programmen erlaubt, Dialoge im Vordergrund zu öffnen. Dazu muß die Windows-API-Funktion "AllowSetForegroundWindow" mit Angabe des Prozeß-Handles oder dem Parameter "aller Prozesse" aufgerufen werden. **Dieser Aufruf ist vor jedem COM-Aufruf sinnvoll, der potentiell Dialoge öffnen könnte.**

In Microsoft Visual Studio befindet sich die Deklaration in WinUser.h.
In Embarcadero Delphi 2009 ist diese Funktion nicht deklariert, dieses muß erst händisch erledigt werden. Der Grund ist die Abwärtskompatibilität von Delphi bis zur Windows Version 95, in der die Funktion noch nicht vorhanden war. In späteren Windows-Versionen eingeführte API-Aufrufe sind deshalb in Delphi nicht deklariert.
In anderen Programmiersprachen und Entwicklungsumgebungen ist entsprechend zu prüfen, ob die Funktion vorhanden ist.
In den SelectLine Rewe-Demos der V. 11.0 ist diese Funktion deklariert und wird beispielhaft aufgerufen. Als Parameter dürfen "alle Prozesse" Dialoge im Vordergrund öffnen.

Im "Spezial-COM-Startmodus" INSTANCING=MULTI funktioniert dieses Verhalten nicht zuverlässig, da der SelectLine COM-Server nicht feststellen kann, von welchem COM-Client der aktuelle Aufruf kommt.

Übersicht über die unterschiedlichen Verhaltensweisen:

COM-Aufruf	Windows-Version	Verhalten	Bemerkungen
Funktion AllowSetForegroundWindow im Windows-API noch nicht vorhanden	Windows 9x Windows NT	Dialoganzeige immer im Vordergrund	Diese Windows-Versionen werden von SelectLine-Programmen nicht mehr unterstützt.
kein Aufruf von AllowSetForegroundWindow	Windows ME alle Versionen ab Windows 2000	blinkendes Symbol in der Taskleiste	Ältere Windows-Versionen werden von SelectLine-Programmen nicht mehr unterstützt.
mit Aufruf von AllowSetForegroundWindow	Windows ME alle Versionen ab Windows 2000	Dialoganzeige immer im Vordergrund	Ältere Windows-Versionen werden von SelectLine-Programmen nicht mehr unterstützt.

siehe auch:
- Delphi-Demo ab V. 11.0 (main.pas)
- MSDN-Hilfe zur Funktion AllowSetForegroundWindow

11. Anhang

1. Erläuterung des Datentyps Variant

Manche Funktion können mit variantem Datentyp aufgerufen werden oder liefern einen solchen zurück.
Varianten können ganze oder reelle Zahlen, Stringwerte, Boolesche Werte, Datum-Uhrzeit-Werte sowie OLE-Automatisierungsobjekte enthalten.

1. Liste der wichtigsten Tabellenschlüssel

Schlüssel	Tabellenname	Beschreibung
BA	BANKEN.DB	Stammdaten Banken
BT	BTXT.DB	Die Buchungstexte aller Buchungen. 1:1 zu BUCH.DB
BU	BUCH.DB	Buchungen
BX	BUCHTEXT.DB	Stammdaten Buchungstexte
FA	FINAMT.DB	Stammdaten Finanzämter
KB	KASSBANK.DB	Stammdaten Bankbezüge
KO	KONTEN.DB	Stammdaten Konten
KU	KUNDEN.DB	Stammdaten Debitoren
LI	LIEFER.DB	Stammdaten Kreditoren
US	USTSCHL.DB	Stammdaten Umsatzsteuer
UD	USDETAIL.DB	Stammdaten Umsatzsteuer (Detailtabelle)
WA	WAEHR.DB	Stammdaten Währungen
WP	WAEHRP.DB	Stammdaten Währungen (Detailtabelle)
ZB	ZAHLBED.DB	Stammdaten Zahlungsbedingungen

Hinweis:
Eine vollständige Liste der Tabellenschlüssel kann in der Rewe unter dem Programmpunkt „Extras | Vorgaben" eingesehen werden.

2. Parameterformate für XFibu.LiesOptionen

Parameter	Beschreibung	ab Rewe Version	Beispiel
System.Version.Hauptversion System.Version.Unterversion1 System.Version.Unterversion2 System.Version.Unterversion3	Auslesen der einzelnen Bestandteile der Versionsnummer.	11.0	
Mandant.Option.<Feldname>	Auslesen des internen Mandantenobjektes. Dieser	11.0	mandant.option.steueridentnr ->Auslesen der SteuerIdentNr (im Programm: Mandanteneinstellungen)

	Parameter entspricht funktional exakt der COM-Funktion XFibu.GetMandField. Aus Performance-Gründen wurde die Funktionalität hier gedoppelt, um getrennte COM-Aufrufe zu vermeiden. Achtung! Da diese Funktion interne Objekte anspricht, kann nicht garantiert werden, daß in späteren Versionen diese Optionen noch ausgelesen werden können.		
Mandant.INI.<Dateiname>.<Block>.<Eintrag>	Liest genau einen Eintrag des angegebenen Blockes aus der angegebenen INI-Datei des Mandanten. Alle INI-Dateien befinden sich in der SQL-Version in der Tabelle INIFILES. Der Dateiname _muß_ ohne Dateierweiterung angegeben werden, d. h. ".INI" entfällt.	11.0	Mandant.ini.mandant.mandant.email ->Email-Adresse aus Mandanteneinstellungen ->liest aus Mandant-INIFILES: MANDANT.INI, Block [Mandant], Eintrag Email mandant.ini.mandant.rewe.version ->Versionsinformation als String ->liest aus Mandant-INIFILES: MANDANT.INI, Block [REWE], Eintrag Version
Mandant.INI.<Dateiname>.<Block>	Liest alle	11.0	Mandant.ini.dbversion.table

	Einträge des angegebenen Blockes aus der angegebenen INI-Datei des Mandanten. Bei Verwendung dieser Parameterform enthält "Optionen" bei der Rückgabe normalerweise mehr Zeilen als bei der Übergabe. Eine direktes Auslesen eines Wertes anhand des Listenindex ist damit nicht mehr möglich.		->Rückgabe aller internen Tabellenversionsnummern ->liest aus Mandant-INIFILES: DBVERSION.INI, Block [Table]
Jahr.INI.<Dateiname>.<Block>.<Eintrag>	Liest genau einen Eintrag aus der INIFILES-Tabelle der aktuellen Jahres-Datenbank.	11.0	jahr.ini.jahr.mandant.kopiermandant ->Auslesen des Vorlagemandanten ->liest aus akt. Jahres-INIFILES: JAHR.INI, Block [Mandant], Eintrag Kopiermandant
Jahr.INI.<Dateiname>.<Block>	Liest alle Einträge des Blockes aus der INIFILES-Tabelle der aktuellen Jahres-Datenbank.	11.0	jahr.ini.jahr.mandant ->Rückgabe aller jahresspezifischen Mandanteneinstellungen ->liest aus akt. Jahres-INIFILES: JAHR.INI, Block [Mandant]
[<JahresDB>].INI.<Dateiname>.<Block>.<Eintrag>	Liest genau einen Eintrag aus der INIFILES-Tabelle der angegebenen Jahres-Datenbank. Mit dieser Parameterform können Einstellungen nicht-aktueller Jahres-Datenbanken ausgelesen	11.0	[200801].ini.jahr.mandant.ABSCHLUSSPERIODEN ->Auslesen des Jahresabschlusses ->liest aus INIFILES der Jahres-Datenbank 200801: JAHR.INI, Block [Mandant], Eintrag Abschlussperioden

	werden. Mit der Funktion XFibu.GetJahrByDate kann der Name der Jahres-Datenbank ermittelt werden.		
[<JahresDB>].INI.<Dateiname>.<Block>	Liest alle Einträge des Blockes aus der INIFILES-Tabelle der angegebenen Jahres-Datenbank.	11.0	[200801].ini.jahr.mandant ->Rückgabe aller jahresspezifischen Mandanteneinstellungen ->liest aus INIFILES der Jahres-Datenbank 200801: JAHR.INI, Block [Mandant]