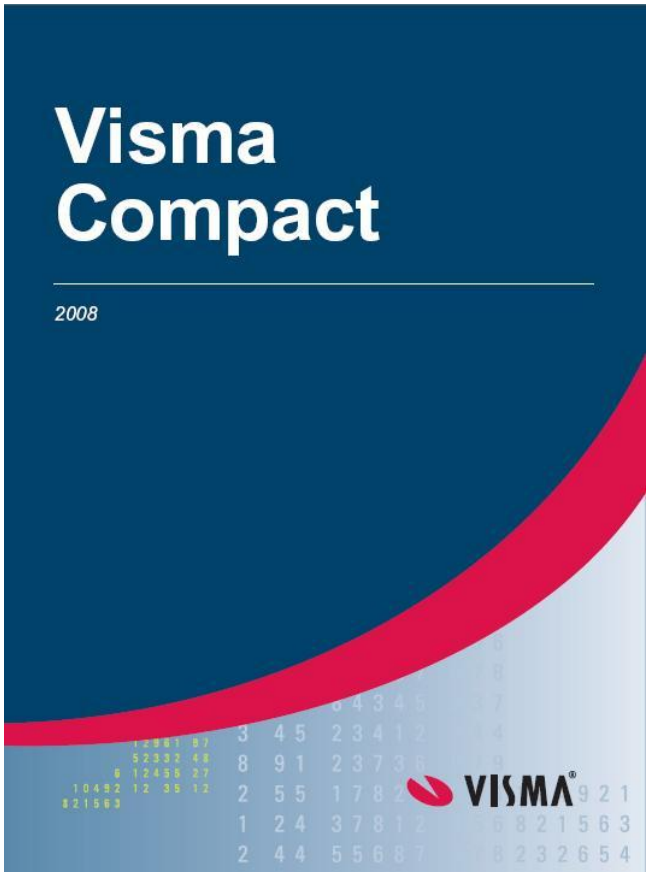
Visma Compact API
Version 6



Referensmanual

Innehållsförteckning

Introduktion

Visma Compact API är ett programmeringsgränssnitt framtaget för att underlätta integrationen av Visma Compact med andra affärssystem samt för att möjliggöra anpassningar av Visma Compact för arbetsuppgifter som normalt inte ingår i Visma Compact. Visma Compact API kan användas i många vanligt förekommande utvecklingsmiljöer och gör det enkelt att skapa t ex fakturor och ordrar med samma affärsregler som Visma Compact följer.

Installation

Visma Compact API distribueras normalt som ett zip-paket. Detta kan packas upp var som helst i filsystemet. I paketet ingår dokumentation, import-filer för de vanligaste utvecklings-miljöerna samt exempelkod. Alla filer är ”native” win32-dller. För att vara så flexibla som möjligt använder alla API-funktionerna Windows standard anropskonventioner. Inga objekt behöver registreras och inga externa beroenden finns.

Själva APIet består av tre dll-filer:

- C4DLL.DLL, databasmotorn
- XFILES.DLL, Visma Compacts klassbibliotek
- XORBASE5.DLL, export av API-funktionerna

API-funktionerna ligger i Xorbase5.dll som i sin tur är beroende av funktioner i C4DLL och XFILES. Alla dessa filer måste alltså finnas tillgängliga för API-programmet. Aktuella versioner av dessa tre filer finns i Visma Compacts programkatalog. För att kunna använda funktionerna i APIet måste också funktionerna deklareras så att API-programmet vet hur de skall anropas. Detta görs lite olika beroende på vilken utvecklingsmiljö man använder.

Programmering

Innan man börjar programmera bör man också förvissa sig om att man har Visma Compact installerat. Enklast är det att ha Visma Compact installerat på samma maskin som används för utvecklingen men det går också att arbeta mot en server förutsatt att servern har Visma Compacts företagsdatabaser åtkomliga.
Om man inte har tillgång till Visma Compact kan man hämta en demo-version från Visma Spcs hemsida.

VB.Net

Importfilen för VB.Net heter Xorbase5.vb. Denna fil läggs enklast till efter att ett projekt skapas genom att man använder ”Add existing item…” och sedan bläddrar sig till filen. För att kunna exekvera måste filerna C4DLL.DLL, XFILES.DLL och XORBASE5.DLL göras tillgängliga för applikationen. Detta kan t ex göras genom att de kopieras till den sökväg som anges i ”Build output path” under projektets kompilerings inställningar.

Delphi

Projekt utvecklade i Delphi måste använda sig av importfilen Xorbase5.pas. Denna läggs till med hjälp av kommandot ”Add file to project...”. Enhetsnamnet är xorbase5. För att kunna

exekvera applikationen måste filerna C4DLL.DLL, XFILES.DLL och XORBASE5.DLL göras tillgängliga.

Visual Basic 6

De projekt som fortfarande använder sig av Visual Basic 6 måste använda importfilen xorbase5.bas. Denna lägger man till sitt projekt t ex genom kommandot ”Add module”. För att kunna exekvera applikationen måste filerna C4DLL.DLL, XFILES.DLL och XORBASE5.DLL göras tillgängliga.

C/C++

API-programmering med C eller C++ är möjlig genom att inkludera importfilen xorbase5.h samt gör filerna filerna C4DLL.DLL, XFILES.DLL och XORBASE5.DLL tillgängliga. Filen Xorbase5.lib innehåller de funktionsdefinitioner som behövs för länkning.

Att ansluta till en företagsdatabas

Visma Compact organiserar data per företag. Varje företag har sin egen unika databas och man kan bara ansluta till ett företag i taget via APIet. Om Visma Compact redan är installerat på maskinen kommer APIet automatiskt att hämta rätt sökväg till databasen. Detta gäller även om databasen ligger på en server.
I de fall där Visma Compact inte är installerat på den dator som kör API-applikationen måste sökvägen till databasen sättas explicit med hjälp av DatabaseSetPath. T ex:

```
DatabaseSetPath("¥¥server¥data¥Compact5¥Företag¥1010")
```

Notera att hela sökvägen, inklusive företagsnamn, måste vara med. DatabaseSetPath ändrar inte sökvägen permanent utan gäller bara för den sessionen.

Visa alla tillgängliga företag

För att hämta företagsnummer för alla företag som är tillgängliga finns två hjälpfunktioner. Med hjälp av dessa kan man iterera sig igenom företagslistan. En datastruktur av typen ClientNameType (eller ClientName) innehåller ett företags namn och nummer efter varje lyckat funktionsanrop. Se exempel nedan:

```
Dim result As Boolean
Dim client As New ClientNameType
Dim item As ListViewItem

result = ClientListInit(client) > 0

While result
        item = New ListViewItem(client.id)
       item.Name = client.id
       item.SubItems.Add(client.name)
       lvClients.Items.Add(item)
       result = ClientListNext(client) > 0
End While
```

Årshantering och datum

Räkenskapsåren i Visma Compact definieras i tabellen Year. I denna tabell anges start- och slutdatum för varje räkenskapsår samt det id-nummer som sedan används i övriga tabeller för att ange i vilket räkenskapsår datat ligger. Om man behöver avgöra i vilket räkenskapsår ett datum infaller kan man använda funktionen `GetYearNo(Date d)` där datum anges som ett julianskt datum. Funktionen returnerar det id-nummer som bestämmer räkenskapsåret vilket inte är detsamma som årtalet.

Datum

Alla datum som används internt i Visma Compact är av typen julianskt datum. Detta är ett heltal där första dagen är 1980-01-01 och dagar därefter är konsekutivt uppräknade. Den 10 januari 2007 är alltså dag 9872. För att omvandla mellan olika datumrepresentationer finns funktionerna YMDToDate(), Year(), Month() och Day(). Se exempel nedan.

```
Public Function DateTimeToXORJulDate(ByVal d As DateTime) As System.Int16
        Return YMDToDate(d.Year, d.Month, d.Day, 0)
End Function

Public Function XORJulDateToDateTime(ByVal d As System.Int16) As DateTime
        Return New System.DateTime(Year(d), Month(d), Day(d))
End Function
```

Seriehantering

Löpnummer för att identifiera poster som verifikat, fakturor och reskontra hanteras via löpnummerserier som definieras i tabellen Series. I version 5.0 av Visma Compact var alla serier kopplade till årshanteringen. För varje räkenskapsår skapade alltså nya fakturaserier.

Från och med version 5.1 är fakturaserierna inte längre bundna till räkenskapsåret. Det innebär att den del av primärnyckeln som har fältnamnet ”YEARNO” numera enbart identifierar en serie och den är inte längre knuten till datum. Denna ändring gäller tabellerna Invoice, InvoiceRow och Ledger.

Distribution

För att en slutanvändare skall kunna köra ett program utvecklat mot Visma Compact API krävs, förutom en installation av Visma Compact, de tre filer som utgör APIet: C4DLL.DLL, XFILES.DLL, XORBASE5.DLL. Inga objekt behöver registreras.
Alla installationer av Visma Compact innehåller de binärfiler som behövs för att applikationer utvecklade mot APIet skall kunna köras. Som API-utvecklare kan man välja mellan att använda de filer som redan finns i installationen eller att packa med API-filer med det egna systemet. Bägge metoderna har sina för- och nackdelar.
Vi rekommenderar dock att man använder filerna som finns i Visma Compacts programkatalog. Man kan använda Windows-API-funktionen SetDllDirectory (finns supportdokument som beskriver detta).

Arbeta med data

Många av API-funktionerna lägger automatiskt upp data utan att man behöver arbeta mot enskilda tabeller och fält. I en del fall kan man emellertid behöva komplettera poster eller kanske utföra uppgifter som inte stöds direkt. Till exempel kan man behöva komplettera kunduppgifter efter att man lagt till en ny kund.
Som API-programmerare har man tillgång till alla fält och tabeller i databasen. Metoden bygger på att man skapar en pekare till de fält man vill arbeta med och använder sedan pekaren då man skriver eller läser data från aktuell post. Positioneringsfunktioner såsom RecordFind, RecordFirst och RecordNext används för att positionera tabellen på den post man är intresserad av. Alla fält och tabeller finns beskrivna i dokumentet ”databasbeskrivning”.

Exempel: Lägg till ny kund

I exemplet nedan läggs en ny kund upp med funktionen PersonAdd. För att kunna komplettera posten skapas en pekare till fältet ”WEB” i tabellen PERSON med hjälp av funktionen RecordField. Denna pekare kan sedan användas antingen för att läsa eller skriva data till fältet.
Efter PersonAdd är tabellen positionerad på den nya posten vilket också betyder att den är aktuell post. För att uppdatera den aktuella posten används funktionen FieldAssignString. Efter att aktuell post har uppdaterats måste värdet returneras till databasen. Detta sker med hjälp av funktionen RecordUpdate.

VB.Net

```
    Private Sub AddCustomer()

        If DatabaseOpen("1010") = DbOk Then
            Dim web As IntPtr = RecordField(PERSON, "WEB")

            If PersonAdd(CUSTOM, "123456", "Alfred Nyman", "", ↩
                         "Storgatan 12", "Box 123", "222 44 Köpingsbruk",↩
                         "", "048-112233", "048-112234", "Kurt")↩
                         = DbOk Then

                FieldAssignString(web, "www.alfrednyman.se")
                RecordUpdate(PERSON)

            End If
            DatabaseClose()
        End If

    End Sub
```

Exempel: Lägg till kund och komplettera ett fält.

Exempel: Sök kund

Man kan också enkelt hämta data från databasen. Man skapar en pekare till det fält i den tabell man är intresserad av. Tabellen positioneras, t ex genom någon sökmetod, och sedan läses värdet ut. I exemplet nedan positioneras tabellen på den post som skapades i exemplet ovan och sedan används FieldValueString för att hämta data. Fältlängder och typ finns i angivet i databasbeskrivningen.

VB.Net

```
    Private Sub FindCustomer()

        If DatabaseOpen("1010") = DbOk Then
            Dim name As New System.Text.StringBuilder(32)
            Dim ptrName As IntPtr = RecordField(PERSON, "PNAME")

            If PersonFind(CUSTOM, "123456") Then
                FieldValueString(ptrName, name, 32)
                MessageBox.Show("Kundens namn: " + name.ToString())
            Else
                MessageBox.Show("Kunde inte hitta kunden!")
            End If
            DatabaseClose()
        End If
```

Exempel: Sök kund.

Skapa reskontraposter

I många fall sköts fakturering i ett försystem men man vill använda Visma Compacts reskontrahantering för redovisning och betalningsbevakning. I dessa fall kan man använda någon av funktionerna LedgerAddCustomerInvoice eller LedgerAddSupplierInvoice beroende på om det är kund- eller leverantörsfakturor.
Båda dessa funktioner innehåller mycket affärsregler. Kund- och leverantörsstatistik uppdateras automatiskt och funktionerna kan även automatiskt utföra bokningen i redovisningen. Se exempel nedan.

VB.Net

```
If DatabaseOpen("1010") = DbOk Then

   If LedgerAddCustomerInvoice("100", 2006, 11, 30, 1000, ↵
         250, FA, 0, 1, "") > 0
   Then
      MessageBox.Show("Wrote ledger to database.")
   Else
      MessageBox.Show("Could not write ledger!")
   End If

   DatabaseClose()
End If
```

Exempel: Lägg till en kundreskontrapost

Funktionsreferens

DatabaseOpen

```
C
        int WINAPI DatabaseOpen( LPCSTR clientNumber );

Visual Basic.Net
        Declare Ansi Function DatabaseOpen Lib "XORBASE5.DLL" ↩
        (ByVal ClientNumber As String) As Integer
```

Öppna företagets databas. Sökvägen till databasen hämtas automatiskt från Visma Compacts installation. Denna funktion måste anropas innan några andra API-funktioner kan användas. Man kan bara ha en instans av databasen öppen åt gången och man bör inte hålla databasen öppen längre än nödvändigt för att undvika prestandaproblem.

Parametrar

clientNumber	Företagets nummer

Returnerar

DbOk	Databasen öppnades utan fel
-1	Sökvägen till databasen är fel
-2	Databasen måste frisläppas
-3	Fel version, databasen måste uppdateras
-4	Fel version, APIet måste uppdateras
-5	Databasen redan öppen

Exempel

```
If DatabaseOpen("4455") = DbOk Then

        If VoucherOpen(0, 2006, 6, 20) > 0 Then

                VoucherAddRow("3011", "Invoice 11", "", "", -100.00)
                VoucherAddRow("1910", "Invoice 11", "", "", 100.00)

                If VoucherClose() > 0 Then
                        MessageBox.Show("Wrote voucher to database.")
                Else
                        MessageBox.Show("Could not write voucher.")
                End If

        End If
        DatabaseClose()
End If
```

DatabaseClose

```
C
	void WINAPI DatabaseClose();

Visual Basic.Net
	Declare Ansi Sub DatabaseClose Lib "XORBASE5.DLL" ()
```

Stänger den instans av databasen som tidigare öppnats. Frigör automatiskt de resurser som allokerats av andra API-funktioner.

Parametrar

(inga)

Returnerar

(inget)

DatabaseSetPath

```
C
      void WINAPI DatabaseSetPath( LPCSTR path );

Visual Basic.Net
      Declare Ansi Sub DatabaseSetPath Lib "XORBASE5.DLL" ⮐
      (ByVal Path As String)
```

Anger en fullständig sökväg till databasens tabeller.

Parametrar

path	Fullständig sökväg

Returnerar

(inget)

Anmärkning

Normalt behöver sökvägen till databasen inte anges då APIet automatiskt hämtar rätt sökväg. Undantagsvis eller för tester kan emellertid denna funktion användas. Sökvägen sparas inte utan gäller bara denna session.

DatabaseGetPath

```
C
      int WINAPI DatabaseGetPath( LPSTR path, int maxPathSize );

Visual Basic.Net
      Declare Ansi Function DatabaseGetPath Lib "XORBASE5.DLL" ↩
      (ByVal Path As StringBuilder, ByVal StringSize As Integer) As Integer
```

Returnerar den fullständiga sökvägen till databastabellerna om en databas är öppen. Om ingen databas är öppen returneras sökvägen till baskatalogen för databaser.

Parametrar

path	Buffert som tar emot sökvägen
maxPathSize	Storlek på bufferten

Returnerar

DbOk	Alltid

DatabaseTableCount

```
C
        int WINAPI DatabaseTableCount();

Visual Basic.Net
        Declare Ansi Function DatabaseTableCount Lib "XORBASE5.DLL" ↵
        () As Integer
```

Returnerar antalet tabeller i databasen.

Parametrar

(inga)

Returnerar

> 0	Antalet tabeller

SetErrorHandler

```
C
        void WINAPI SetErrorHandler ⮐
        ( int (CALLBACK *ErrorCallback)(int, int, char *) );

Visual Basic.Net
        Declare Ansi Sub SetErrorHandler Lib "XORBASE5.DLL" ⮐
        (ByVal ErrorCallback As ErrorHandler)
```

Anger en felhanterare som anropas då ett fel uppstår i databasmotorn.

Parametrar

ErrorCallback	En pekare till en callback funktion

Returnerar

(inget)

Anmärkning

De flesta fel som returneras av databasmotorn kan anses vara fatala. Därför bör man avsluta exekveringen efter att felet hanterats.

TableFieldCount

```
C
      int WINAPI TableFieldCount( int tableIndex);

Visual Basic.Net
      Declare Ansi Function TableFieldCount Lib "XORBASE5.DLL" ↵
      (ByVal TableIndex As Integer) As Integer
```

Returnerar antalet fält i angiven tabell.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

> 0	Antalet fält
-1	Databasen är inte öppen

TableRecordCount

C

```
int WINAPI TableRecordCount( int tableIndex);
```

Visual Basic.Net

```
Declare Ansi Function TableRecordCount Lib "XORBASE5.DLL" ↩
(ByVal TableIndex As Integer) As Integer
```

Returnerar antal poster i angiven tabell.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

> 0	Antalet poster
-1	Databasen är inte öppen

TableName

```
C
      int WINAPI TableName( int tableIndex, LPSTR buffer, int bufferSize );

Visual Basic.Net
      Declare Ansi Function TableName Lib "XORBASE5.DLL" ⮐
      (ByVal Path As StringBuilder, ByVal StringSize As Integer) As Integer
```

Hämtar tabellens namn baserat på ordningsnummer.

Parametrar

tableIndex	Tabellens ordningsnummer
buffer	Buffert som tar emot tabellens namn
bufferSize	Storlek på bufferten

Returnerar

DbOk	Operationen lyckades
-1	Databasen är inte öppen

RecordCreate

C

```
int WINAPI RecordCreate( int tableIndex);
```

Visual Basic.Net

```
Declare Ansi Function RecordCreate Lib "XORBASE5.DLL" ↩
(ByVal TableIndex As Integer) As Integer
```

Positionerar tabellen på en ny blank post.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

DbOk	Operationen lyckades
-1	Databasen är inte öppen
-2	Positioneringen misslyckades

Anmärkning

Funktionen används ihop med RecordAdd.

Exempel

```
void AddArticleLow()
{
	Field artNo = RecordField( ARTICLE, "ARTNO");
	Field artName = RecordField( ARTICLE, "ARTNAME");
	Field price = RecordField( ARTICLE, "ARTPRICE0");

	RecordCreate(ARTICLE);

	FieldAssignString( artNo, "99188");
	FieldAssignString( artName, "Skruv");
	FieldAssignDouble( price, 12.25);

	RecordAdd(ARTICLE);
}
```

RecordFind

C

```
int WINAPI RecordFind( Key key);
```

Visual Basic.Net

```
Declare Ansi Function RecordFind Lib "XORBASE5.DLL" ↩
(ByVal Key As IntPtr) As Integer
```

Söker efter en post baserat på en nyckel.

Parametrar

key	Pekare till en nyckel som returnerats av RecordKey

Returnerar

<> 0	Posten hittades
0	Posten kunde inte hittas

Exempel

```
void FindArticle()
{
    char artName[64];
    Key kArticle = RecordKey(ARTICLE, 1);
    Field kArtNo = KeyField(kArticle, "ARTNO");

    Field fArtName = RecordField(ARTICLE, "ARTNAME");

    FieldAssignString(kArtNo, "1002");

    if( RecordFind(kArticle) != 0 )
    {
        FieldValueString(fArtName, artName, 64);
        cout << artName << endl;
    }
}
```

RecordFirst

```
C
      int WINAPI RecordFirst( int tableIndex, int keyNumber);

Visual Basic.Net
      Declare Ansi Function RecordFirst Lib "XORBASE5.DLL" ↩
      (ByVal TableIndex As Integer, ByVal KeyNumber As Integer) As Integer
```

Positionerar aktuell post på första posten med sortering angiven av keyNumber.

Parametrar

tableIndex	Tabellens ordningsnummer
keyNumber	Nyckel i tabellen

Returnerar

DbOk	Operationen lyckades
-1	Databasen är inte öppen
-2	Positioneringen misslyckades

RecordNext

C

```
int WINAPI RecordNext( int tableIndex);
```

Visual Basic.Net

```
Declare Ansi Function RecordNext Lib "XORBASE5.DLL" ↩
(ByVal TableIndex As Integer) As Integer
```

Positionerar nästa post som aktuell post.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

DbOk	Operationen lyckades
-1	Databasen är inte öppen
-2	Positioneringen misslyckades

RecordPrev

C

```
int WINAPI RecordPrev( int tableIndex);
```

Visual Basic.Net

```
Declare Ansi Function RecordPrev Lib "XORBASE5.DLL" ↩
(ByVal TableIndex As Integer) As Integer
```

Positionerar föregående post som aktuell post.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

DbOk	Operationen lyckades
-1	Databasen är inte öppen
-2	Positioneringen misslyckades

RecordLast

```
C
      int WINAPI RecordLast( int tableIndex, int keyNumber);

Visual Basic.Net
      Declare Ansi Function RecordLast Lib "XORBASE5.DLL" ↩
      (ByVal TableIndex As Integer, ByVal KeyNumber As Integer) As Integer
```

Positionerar aktuell post på sista posten med sortering angiven av keyNumber.

Parametrar

tableIndex	Tabellens ordningsnummer
keyNumber	Nyckel i tabellen

Returnerar

DbOk	Operationen lyckades
-1	Databasen är inte öppen
-2	Positioneringen misslyckades

RecordUpdate

```
C
      int WINAPI RecordUpdate( int tableIndex);

Visual Basic.Net
      Declare Ansi Function RecordUpdate Lib "XORBASE5.DLL" ↩
      (ByVal TableIndex As Integer) As Integer
```

Uppdaterar aktuell post till en databastabell.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

DbOk	Posten sparades utan fel
-1	Databasen är inte öppen
-2	Posten kunde inte uppdateras

Exempel

```
void AddPerson()
{
    Field web = RecordField( PERSON, "WEB");

    if( PersonAdd( CUSTOM, "12345", "Alfred Nyman", "", "Storgatan 12",
                   "Box 123", "222 44 Köpingsbruk", "", "048-112233",
                    "048-112232", "Kurt", -1, -1, -1) > -1 )
    {
        FieldAssignString( web, "www.alfrednyman.se");
        RecordUpdate(PERSON);
    }
}
```

RecordAdd

```
C
        int WINAPI RecordAdd( int tableIndex);

Visual Basic.Net
        Declare Ansi Function RecordAdd Lib "XORBASE5.DLL" ⮐
        (ByVal TableIndex As Integer) As Integer
```

Lägger till aktuell post till en databastabell.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

DbOk	Posten sparades utan fel
-1	Databasen är inte öppen
-2	Posten kunde inte läggas till

Anmärkning
För att lägga till en ny post måste först RecordCreate anropas.

Exempel

Lägg till en artikel med lågnivåfunktioner.

```
C++

void AddArticleLow()
{
        Field artNo = RecordField( ARTICLE, "ARTNO");
        Field artName = RecordField( ARTICLE, "ARTNAME");
        Field price = RecordField( ARTICLE, "ARTPRICE0");

        RecordCreate(ARTICLE);

        FieldAssignString( artNo, "99188");
        FieldAssignString( artName, "Skruv");
        FieldAssignDouble( price, 12.25);

        RecordAdd(ARTICLE);
}
```

RecordDelete

```
C
      int WINAPI RecordDelete( int tableIndex);

Visual Basic.Net
      Declare Ansi Function RecordDelete Lib "XORBASE5.DLL" ↩
      (ByVal TableIndex As Integer) As Integer
```

Raderar aktuell post.

Parametrar

tableIndex	Tabellens ordningsnummer

Returnerar

DbOk	Posten raderades utan fel
-1	Databasen är inte öppen
-2	Posten kunde inte raderas

Anmärkning
RecordDelete opererar direkt på tabellen. Oförsiktigt användande kan därför riskera databasintegriteten.

RecordField

```
C
      Field WINAPI RecordField( int tableIndex, LPCSTR fieldName);

Visual Basic.Net
      Declare Ansi Function RecordField Lib "XORBASE5.DLL" ⮐
      (ByVal TableIndex As Integer, ByVal FieldName As String) As IntPtr
```

Hämtar en pekare till ett fält i angiven tabell. Eftersom funktionen allokerar lite minne bör man undvika att anropa funktionen inom en loop. Det allokerade minnet frigörs vid DatabaseClose.

Parametrar

tableIndex	Tabellens ordningsnummer
fieldName	Fältnamn i tabellen

Returnerar

<>0	Pekare till ett fält
0	Fältnamnet finns inte i denna tabell

Anmärkning
För definitioner av tabeller, fält och nycklar se databasbeskrivningen.

RecordKey

```
C
        Key WINAPI RecordKey( int tableIndex, int keyNumber);

Visual Basic.Net
        Declare Ansi Function RecordKey Lib "XORBASE5.DLL" ⮐
        (ByVal TableIndex As Integer, ByVal KeyNumber As Integer) As Integer
```

Hämtar en pekare till en nyckel för angiven tabell. Eftersom funktionen allokerar lite minne bör man undvika att anropa funktionen inom en loop. Det allokerade minnet frigörs vid DatabaseClose.

Parametrar

tableIndex	Tabellens ordningsnummer
keyNumber	Nyckel i tabellen

Returnerar

<>0	Pekare till en nyckel
0	Nyckeln existerar ej

Anmärkning
För definitioner av tabeller, fält och nycklar se databasbeskrivningen.

Exempel

```
void FindArticle()
{
    char artName[64];
    Key kArticle = RecordKey(ARTICLE, 1);
    Field kArtNo = KeyField(kArticle, "ARTNO");

    Field fArtName = RecordField(ARTICLE, "ARTNAME");

    FieldAssignString(kArtNo, "1002");

    if( RecordFind(kArticle) != 0 )
    {
        FieldValueString(fArtName, artName, 64);
        cout << artName << endl;
    }
}
```

KeyField

```
C
      Field WINAPI KeyField( Key key, LPCSTR fieldName);

Visual Basic.Net
      Declare Ansi Function KeyField Lib "XORBASE5.DLL" ⮐
      (ByVal Key As IntPtr, ByVal FieldName As String) As IntPtr
```

Hämtar ett nyckelfält baserat på angivet fält inom angiven nyckel. Eftersom funktionen allokerar lite minne bör man undvika att anropa funktionen inom en loop. Det allokerade minnet frigörs vid DatabaseClose.

Parametrar

key	Nyckel som returnerats från RecordKey

Returnerar

<> 0	Pekare till ett fält
0	Fältet kunde inte hittas

Anmärkning

För definitioner av tabeller, fält och nycklar se databasbeskrivningen.

Exempel

```
void FindArticle()
{
    char artName[64];
    Key kArticle = RecordKey(ARTICLE, 1);
    Field kArtNo = KeyField(kArticle, "ARTNO");

    Field fArtName = RecordField(ARTICLE, "ARTNAME");

    FieldAssignString(kArtNo, "1002");

    if( RecordFind(kArticle) != 0 )
    {
        FieldValueString(fArtName, artName, 64);
        cout << artName << endl;
    }
}
```

FieldLength

```
C
      int WINAPI FieldLength( Field field);

Visual Basic.Net
      Declare Ansi Function FieldLength Lib "XORBASE5.DLL" ⮠
      (ByVal Field As IntPtr) As Integer
```

Hämtar fältlängden för angivet fält.

Parametrar

field	Pekare som returnerats från RecordField

Returnerar

> 0	Fältets längd

FieldType

```
C
      int WINAPI FieldType( Field field);

Visual Basic.Net
      Declare Ansi Function FieldType Lib "XORBASE5.DLL" ⮐
      (ByVal Field As IntPtr) As Integer
```

Hämtar typen av ett fält. Typen representeras av ASCII-värdet för en teckenkonstant. Vanligen kan man avgöra vilken typ fältet representerar med hjälp av databasbeskrivningen. Denna funktion kan användas om man vill hantera fälten dynamiskt.

Parametrar

field	Pekare som returnerats från RecordField

Returnerar

> 0	Fältets typ:
66	Flyttal, 'B'
67	Textsträng, 'C'
68	Datum, 'D'
73	Heltal, 'I'
76	Boolean, 'L'

FieldName

```
C
      int WINAPI FieldName( int tableIndex, int fieldIndex, ↩
      LPSTR buffer, int bufferSize);

Visual Basic.Net
      Declare Ansi Function FieldName Lib "XORBASE5.DLL" ↩
      (ByVal TableIndex As Integer, ByVal FieldIndex As Integer, ↩
      ByVal StringBuffer As StringBuilder, ByVal StringSize As Integer) ↩
      As Integer
```

Hämtar namnet för ett angivet fält i en tabell.

Parametrar

tableIndex	Tabellens ordningsnummer
fieldIndex	Fältets ordningsnummer inom tabellen
buffer	Buffert som tar emot namnet
bufferSize	Storlek på bufferten

Returnerar

1	Namnet hämtades utan fel
0	Databasen är inte öppen

FieldAssignDouble

```
C
      int WINAPI FieldAssignDouble( Field field, double value);

Visual Basic.Net
      Declare Ansi Function FieldAssignDouble Lib "XORBASE5.DLL" ↵
      (ByVal Field As IntPtr, ByVal value As Double) As Integer
```

Tilldela angivet fält ett flyttalsvärde.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField
value	Ett flyttalsvärde

Returnerar

(inget)

FieldAssignLong

```
C
      int WINAPI FieldAssignLong( Field field, int value);

Visual Basic.Net
      Declare Ansi Function FieldAssignLong Lib "XORBASE5.DLL" ↩
      (ByVal Field As IntPtr, ByVal value As Integer) As Integer
```

Tilldela angivet fält ett heltal.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField
value	Ett heltal

Returnerar

(inget)

FieldAssignString

```
C
      int WINAPI FieldAssignString( Field field, LPCSTR string);

Visual Basic.Net
      Declare Ansi Function FieldAssignString Lib "XORBASE5.DLL" ↵
      (ByVal Field As IntPtr, ByVal value As String) As Integer
```

Tilldela angivet fält en textsträng.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField
string	En textsträng

Returnerar

(inget)

Exempel

```
C++

void AddPerson()
{
    Field web = RecordField( PERSON, "WEB");

    if( PersonAdd( CUSTOM, "12345", "Alfred Nyman", "", "Storgatan 12",
                   "Box 123", "222 44 Köpingsbruk", "", "048-112233",
                    "048-112232", "Kurt", -1, -1, -1) > -1 )
    {
        FieldAssignString( web, "www.alfrednyman.se");
        RecordUpdate(PERSON);
    }
}
```

FieldAssignBool

```
C
      int WINAPI FieldAssignBool( Field field, bool value);

Visual Basic.Net
      Declare Ansi Function FieldAssignBool Lib "XORBASE5.DLL" ↩
      (ByVal Field As IntPtr, ByVal value As Boolean) As Integer
```

Tilldela angivet fält ett logiskt värde.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField
value	Sant eller falskt

Returnerar

(inget)

FieldAssignDate

```
C
      int WINAPI FieldAssignDate( Field field, Date value);

Visual Basic.Net
      Declare Ansi Function FieldAssignDate Lib "XORBASE5.DLL" ↩
      (ByVal Field As IntPtr, ByVal value As Short) As Integer
```

Tilldela angivet fält ett julianskt datum.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField
value	Ett julianskt datum

Returnerar

(inget)

FieldValueDouble

```
C
      double WINAPI FieldValueDouble( Field field);

Visual Basic.Net
      Declare Ansi Function FieldValueDouble Lib "XORBASE5.DLL" ↵
      (ByVal Field As IntPtr) As Double
```

Hämtar ett flyttalsvärde från angivet fält.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField

Returnerar

Ett flyttalsvärde

FieldValueLong

C

```
int WINAPI FieldValueLong( Field field);
```

Visual Basic.Net

```
Declare Ansi Function FieldValueLong Lib "XORBASE5.DLL" ↵
(ByVal Field As IntPtr) As Integer
```

Hämtar ett heltal från angivet fält.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField

Returnerar

Ett heltal

FieldValueString

```
C
      int WINAPI FieldValueString( Field field, ↵
      LPSTR buffer, int bufferSize);

Visual Basic.Net
      Declare Ansi Function FieldValueString Lib "XORBASE5.DLL" ↵
      (ByVal Field As IntPtr, ByVal StringBuffer As StringBuilder, ↵
      ByVal StringSize As Integer) As Integer
```

Hämtar en textsträng från angivet fält.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField
buffer	En buffert som tar emot textsträngen
bufferSize	Storlek på bufferten

Returnerar

> 0	Textsträngens längd

FieldValueBool

```
C
      bool WINAPI FieldValueBool( Field field);

Visual Basic.Net
      Declare Ansi Function FieldValueBool Lib "XORBASE5.DLL" ↩
      (ByVal Field As IntPtr) As Boolean
```

Hämtar ett logiskt värde från angivet fält.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField

Returnerar

Ett logiskt värde

FieldValueDate

```
C
      Date WINAPI FieldValueDate( Field field);

Visual Basic.Net
      Declare Ansi Function FieldValueDate Lib "XORBASE5.DLL" ↩
      (ByVal Field As IntPtr) As Short
```

Hämtar ett julianskt datum från angivet fält.

Parametrar

field	Pekare som returnerats från RecordField eller KeyField

Returnerar

Ett julianskt datum

InvoiceFind

```
C
      int WINAPI InvoiceFind(int serieNumber, int invoiceIndex, ↵
      int invoiceNumber);

Visual Basic.Net
      Declare Ansi Function InvoiceFind Lib "XORBASE5.DLL"  ↵
      (ByVal serieNumber As Integer, ByVal invoiceIndex As Integer, ↵
      ByVal invoiceNumber As Integer) As Integer
```

Söker upp ett fakturahuvud och sätter det som aktuell post.

Parametrar

serieNumber	Fakturaseriens nummer
invoiceIndex	Dokumenttyp: XORDER = Order XCUSTINVOICE = Kundfaktura XPURCHASE = Beställning
invoiceNumber	Fakturans löpnummer

Returnerar

0	Fakturahuvudet kunde inte hittas
1	Fakturahuvudet hittades

Anmärkning

InvoiceRowFind

```
C
      int WINAPI InvoiceRowFind(int serieNumber, int invoiceIndex, ↵
      int invoiceNumber, int rowNumber);

Visual Basic.Net
      Declare Ansi Function InvoiceRowFind Lib "XORBASE5.DLL"  ↵
      (ByVal serieNumber As Integer, ByVal invoiceIndex As Integer, ↵
      ByVal invoiceNumber As Integer, ByVal rowNumber As Integer) ↵
      As Integer
```

Söker upp en fakturarad och sätter den som aktuell post.

Parametrar

serieNumber	Fakturaseriens nummer
invoiceIndex	Dokumenttyp: XORDER = Order XCUSTINVOICE = Kundfaktura XPURCHASE = Beställning
invoiceNumber	Fakturans löpnummer
rowNumber	Fakturaradnummer (Första raden = 1)

Returnerar

0	Fakturaraden kunde inte hittas
1	Fakturaraden hittades

Anmärkning

InvoiceOpen

```
C
      int WINAPI InvoiceOpen( LPCSTR persNumber, int type, int status, ↩
      int year, int month, int day, double freight, double fee);

Visual Basic.Net
      Declare Ansi Function InvoiceOpen Lib "XORBASE5.DLL" ↩
      (ByVal PersNumber As String, ByVal type As Integer, ↩
      ByVal Status As Integer, ByVal Year As Integer, ↩
      ByVal Month As Integer, ByVal Day As Integer, ↩
      ByVal Freight As Double, ByVal Fee As Double) As Integer
```

Förbereder för att skapa en ny faktura, order eller beställning. Allokerar utrymme och kontrollerar indata.

Parametrar

persNumber	Kund eller leverantörs-ID
type	Fakturatyp: XORDER = Order XCUSTINVOICE = Kundfaktura XPURCHASE = Beställning
status	Fakturastatus: NOTPRINTED = Ej utskriven PRINTED = Utskriven TRANSFERED = Överförd PERMANENT = Permanent NOTREADY = Ej färdig
year	Fakturadatumets år angivet med fyra siffror
month	Fakturadatumets månad
day	Fakturadatumets dag
freight	Fraktkostnad
fee	Expeditionsavgift

Returnerar

> 0	Fakturaseriens nummer
-1	Databasen var inte öppen
-2	Inget räkenskapsår för detta datum
-3	Räkenskapsåret är stängt eller avslutat för detta datum
-4	Personen kunde inte hittas

Anmärkning

Funktionen används tillsammans med InvoiceAddRow och InvoiceClose. Inga data sparas innan InvoiceClose anropas. Om år, månad och dag anges till 0 sätts dagens datum som fakturadatum. Moms på frakt och expeditionsavgift beräknas automatiskt baserat på inställningar i reserverade konton.
För att skapa en kreditfaktura skall alla belopp på fakturan vara negativa.

InvoiceAddRow

C

```
int WINAPI InvoiceAddRow( LPCSTR articleID, LPCSTR articleName, ↵
LPCSTR accountNumber, LPCSTR object1, LPCSTR object2, LPCSTR unit, ↵
double items, double price, double discount);
```

Visual Basic.Net

```
Declare Ansi Function InvoiceAddRow Lib "XORBASE5.DLL" ↵
(ByVal ArticleID As String, ByVal ArticleName As String, ↵
ByVal AccountNumber As String, ByVal Object1 As String, ↵
ByVal Object2 As String, ByVal Unit As String, ↵
ByVal Items As Double, ByVal Price As Double, ↵
ByVal Discount As Double) As Integer
```

Lägger till en fakturarad till den faktura som initierats med föregående anrop av InvoiceOpen.

Parametrar

articleID	ArtikelID
articleName	Artikelnamn eller text för denna rad
accountNumber	Kontonummer för denna fakturarad
object1	Objekt av typ 1
object2	Objekt av typ 2
unit	Enhet
items	Antal artiklar
price	Pris per artikel exklusive moms
discount	Eventuell rabatt, angiven i procent

Returnerar

> 0	Radnummer för denna fakturarad
-1	Databasen är inte öppen
-2	InvoiceOpen har inte anropats
-3	Kontot saknas i databasen
-4	Object1 saknas eller har fel typ
-5	Object2 saknas eller har fel typ
-6	Angiven artikel kan inte hittas

Anmärkning

Funktionen används tillsammans med InvoiceOpen och InvoiceClose. Inga data sparas innan InvoiceClose anropas. En fri textrad, med eller utan belopp, kan skapas genom att utesluta artikelID.

Moms beräknas automatiskt baserat på inställningar för aktuellt försäljningskonto. Om konto och/eller objekt inte anges används i första hand artikelns inställningar, i andra hand personens inställningar och slutligen reserverade konton.

InvoiceClose

```
C
      int WINAPI InvoiceClose();

Visual Basic.Net
      Declare Ansi Function InvoiceClose Lib "XORBASE5.DLL" () As Integer
```

Avslutar och sparar den faktura som initierats av ett föregående anrop till InvoiceOpen.

Parametrar

(inga)

Returnerar

> 0	Fakturans löpnummer
-1	Databasen är inte öppen
-2	InvoiceOpen har inte anropats
-3	Fakturan kunde inte sparas

Anmärkning

InvoiceClose sparar de fakturarader som lagts till av InvoiceRowAdd. Alla rader sparas i en transaktion. Fakturan summeras automatiskt. Om öresavrundning är valt i företags-inställningarna avrundas fakturabeloppet automatiskt.

VoucherFind

```
C
        int WINAPI VoucherFind(int yearNumber, int series, ↩
        int voucherNumber, int rowNumber);

Visual Basic.Net
        Declare Ansi Function VoucherFind Lib "XORBASE5.DLL" ↩
        (ByVal yearNumber As Integer, ByVal series As Integer, ↩
        ByVal voucherNumber As Integer, ByVal rowNumber As Integer) ↩
        As Integer
```

Söker upp en verifikatrad och positionerar den som aktuell post.

Parametrar

yearNumber	Räkenskapsårsnummer
series	Verifikatserie (0 – 9 , 0 = Serie A)
voucherNumber	Verifikatnummer
rowNumber	Radnummer (första radnummer = 1)

Returnerar

0	Verifikatraden kunde inte hittas
1	Verifikatraden hittades

Anmärkning

VoucherOpen

```
C
      int WINAPI VoucherOpen( int series, int year, int month, int day);

Visual Basic.Net
      Declare Ansi Function VoucherOpen Lib "XORBASE5.DLL" ⮐
      (ByVal Series As Integer, ByVal Year As Integer, ⮐
      ByVal Month As Integer, ByVal Day As Integer) As Integer
```

Förbereder för att skapa ett nytt verifikat. Allokerar utrymme och kontrollerar indata.

Parametrar

series	Verifikatserie (0 – 9 , 0 = Serie A)
year	År, angivet med fyra siffror
month	Månad
day	Dag

Returnerar

> 0	Räkenskapsårets ID
-1	Databasen var inte öppen
-2	Inget räkenskapsår för detta datum
-3	Räkenskapsåret är stängt eller avslutat för detta datum
-4	Ogiltlig verifikatserie

Anmärkning

Funktionen används tillsammans med VoucherAddRow och VoucherClose. Inga data sparas innan VoucherClose anropas.

VoucherAddRow

```
C
      int WINAPI VoucherAddRow( LPCSTR accountNumber, ↩
      LPCSTR voucherText, LPCSTR object1, LPCSTR object2, double amount );

Visual Basic.Net
      Declare Ansi Function VoucherAddRow Lib "XORBASE5.DLL" ↩
      (ByVal AccountNumber As String, ByVal VoucherText As String, ↩
      ByVal Object1 As String, ByVal Object2 As String, ↩
      ByVal amount As Double) As Integer
```

Lägger till en verifikatrad till det verifikat som initierats med föregående anrop av VoucherOpen.

Parametrar

accountNumber	Kontonummer för denna rad
voucherText	Text för denna rad. Se anmärkning.
object1	Objekt av typ 1
object2	Objekt av typ 2
amount	Transaktionsbelopp

Returnerar

> 0	Radnummer för denna verifikatrad
-1	Databasen är inte öppen
-2	VoucherOpen har inte anropats
-3	Kontot saknas i databasen
-4	Object1 saknas eller har fel typ
-5	Object2 saknas eller har fel typ

Anmärkning

Funktionen används tillsammans med VoucherOpen och VoucherClose. Inga data sparas innan VoucherClose anropas.
När man bokar ett verifikat med koppling till leverantörs- eller kundreskontrapost med hjälp av LedgerAddSupplierInvoice eller LedgerAddCustomerInvoice kan man utelämna verifikationstexten. Texten skapas då enligt Visma Compacts standardmall.

VoucherClose

```
C
	int WINAPI VoucherClose();

Visual Basic.Net
	Declare Ansi Function VoucherClose Lib "XORBASE5.DLL" () As Integer
```

Avslutar och sparar det verifikat som initierats av ett föregående anrop till VoucherOpen.

Parametrar

(inga)

Returnerar

> 0	Verifikatets nummer
-1	Databasen är inte öppen
-2	VoucherOpen har inte anropats
-3	Verifikatet balanserar inte
-4	Verifikatet kunde inte sparas

Anmärkning

VoucherClose sparar de verifikatrader som lagts till av VoucherRowAdd. Alla rader sparas i en transaktion. För bokning av verifikat i samband med reskontraposter se också funktionen LedgerAddSupplierInvoice.

LedgerFind

```
C
      int WINAPI LedgerFind(int serieNumber, int personType, ↩
      int ledgerType, int ledgerNumber);

Visual Basic.Net
      Declare Ansi Function LedgerFind Lib "XORBASE5.DLL" ↩
      (ByVal serieNumber As Integer, ByVal personType As Integer, ↩
      ByVal ledgerType As Integer, ByVal ledgerNumber As Integer) ↩
      As Integer
```

Letar upp en reskontrapost och positionerar den som aktuell post.

Parametrar

serieNumber	Fakturaseriens nummer
personType	Persontyp: CUSTOM = kundreskontrapost SUPPLY = leverantörsreskontrapost
ledType	Reskontratyp, någon av följande: FA Faktura KN Kreditnota RF Räntefaktura AC A conto post BE Betalning
ledgerNumber	Reskontrapostens löpnummer

Returnerar

0	Ingen post kunde hittas
1	Rekontraposten hittades

Anmärkning

Reskontrapostens löpnummer kan ligga i olika serier, beroende på vilken reskontratyp det är.

LedgerAddCustomerInvoice

```
C
      int WINAPI LedgerAddCustomerInvoice( LPCSTR personNumber, ↩
      int year, int month, int day, double amount, double vat, ↩
      int ledType, int invoiceNumber, int update, LPCSTR account);

Visual Basic.Net
     Declare Ansi Function LedgerAddCustomerInvoice Lib "XORBASE5.DLL" ↩
     (ByVal personNumber As String, ByVal year As Integer, ↩
     ByVal month As Integer, ByVal day As Integer, ↩
     ByVal amount As Double, ByVal vat As Double, ↩
     ByVal ledType As Integer, ByVal invoiceNumber As Integer, ↩
     ByVal update As Integer, ByVal account As String) As Integer
```

Lägger till en kundreskontrapost i Visma Compacts reskontraregister. Personstatistik uppdateras automatiskt. Bokföringen kan uppdateras automatiskt genom att ett verifikat skapas baserat på ingående värden.

Parametrar

personNumber	Kund-ID
year	År, angivet med fyra siffror, för bokföringsdatum
month	månad (1–12)
day	dag (1–31)
amount	Fakturabelopp inklusive moms
vat	Momsbelopp
ledType	Reskontratyp, någon av följande: FA Faktura KN Kreditnota RF Räntefaktura
invoiceNumber	Sätts till 0 om Compacts interna löpnummerräknare skall användas (rekommenderas). Kan annars sättas till eget värde. Se anmärkning.
update	Verifikatkoppling 2 Koppla manuellt verifikat (se anmärkning) 1 Skapa automatiskt verifikat 0 Ingen koppling till verifikat
account	Sätts till tom sträng (””) om kundens försäljningskonto skall användas, annars till önskat försäljningskonto.

Returnerar

> 0	Reskontrapostens fakturanummer
-1	Databasen var inte öppen
-2	Inget räkenskapsår för detta datum
-3	Räkenskapsåret är stängt eller avslutat för detta datum
-4	Ogiltlig reskontratyp
-5	Kunden kunde inte hittas

-6	Angivet fakturanummer kolliderar med Compacts löpnummerserie
-8	Posten kunde inte sparas
-9	Kontot som angavs som parameter saknas
-10	Kunde inte spara kopplat verifikat (update = 2)
-13	Kopplat verifikat balanserar inte

Anmärkning
Om man vill använda en egen serie för fakturanummer måste man se till att denna inte kolliderar med Compacts interna löpnummerserie för kundfakturor. Man måste också se till att det egna fakturanumret räknas upp korrekt.

I många fall vill man uppdatera bokföringen i samband med att man sparar en reskontrapost. Om man gör detta i samband med att man skapar reskontraposten kan man använda Visma Compacts inbyggda affärsregler för att se till att alla poster hänger ihop på ett konsekvent sätt. Om parametern update sätts till 1 skapar funktionen automatiskt ett standardverifikat baserat på ingångsvärdena till funktionen. Ofta kan man dock ha behov av att göra en mer specialicerad bokning och man sätter då update till 2.

Med parametern update satt till 2 kan funktionen användas i stället för VoucherClose för att spara ett verifikat i samma transaktion som reskontraposten samtidigt som alla kopplingar uppdateras automatiskt (se exempel vid LedgerAddSupplierInvoice). I detta fall hanteras också verifikattexten på verifikatraderna något annorlunda. Om man låter parametern voucherText vara blank i funktionen VoucherAddRow kommer Visma Compacts standardtext för kundbokning att användas. Lägger man till egen text så kommer reskontratyp och löpnummer att infogas framför den egna texten.

LedgerAddSupplierInvoice

C

```
int WINAPI LedgerAddSupplierInvoice( LPCSTR personNumber, ↩
LPCSTR suppliersInvoiceNo, int year, int month, int day, ↩
double amount, double vat, ↩
int ledType, int update, LPCSTR account);
```

Visual Basic.Net

```
Declare Ansi Function LedgerAddSupplierInvoice Lib "XORBASE5.DLL" ↩
(ByVal personNumber As String, ByVal suppliersInvoiceNo As String, ↩
ByVal year As Integer, ↩
ByVal month As Integer, ByVal day As Integer, ↩
ByVal amount As Double, ByVal vat As Double, ↩
ByVal ledType As Integer, ↩
ByVal update As Integer, ByVal account As String) As Integer
```

Lägger till en leverantörsreskontrapost i Visma Compacts reskontraregister. Personstatistik uppdateras automatiskt. Bokföringen kan uppdateras automatiskt genom att ett verifikat skapas baserat på ingående värden eller man kan koppla ett manuellt skapat verifikat till reskontraposten (se anmärkning). Ett fakturadokument kan kopplas till denna post om funktionen DocumentAddSupplierInvoice() anropas *innan* denna funktion.

Parametrar

personNumber	Leverantörs-ID
suppliersInvoiceNo	Leverantörens fakturanummer om känt.
year	År, angivet med fyra siffror, för bokföringsdatum
month	månad (1–12)
day	dag (1–31)
amount	Fakturabelopp inklusive moms
vat	Momsbelopp
ledType	Reskontratyp, någon av följande: FA Faktura KN Kreditnota RF Räntefaktura
update	Verifikatkoppling 2 Koppla manuellt verifikat (se anmärkning) 1 Skapa automatiskt verifikat 0 Ingen koppling till verifikat
account	Sätts till tom sträng (””) om leverantörens inköpskonto skall användas, annars till önskat inköpskonto (update = 1).

Returnerar

> 0	Reskontrapostens löpnummer
-1	Databasen var inte öppen
-2	Inget räkenskapsår för detta datum
-3	Räkenskapsåret är stängt eller avslutat för detta datum

-4	Ogiltlig reskontratyp
-5	Leverantören kunde inte hittas
-6	Posten kunde inte sparas
-7	Kunde inte kopiera kopplat dokument
-9	Kontot som angavs som parameter saknas
-10	Kunde inte spara kopplat verifikat (update = 2)
-13	Kopplat verifikat balanserar inte

Anmärkning
I många fall vill man uppdatera bokföringen i samband med att man sparar en reskontrapost. Om man gör detta i samband med att man skapar reskontraposten kan man använda Visma Compacts inbyggda affärsregler för att se till att alla poster hänger ihop på ett konsekvent sätt. Om parametern update sätts till 1 skapar funktionen automatiskt ett standardverifikat baserat på ingångsvärdena till funktionen. Ofta kan man dock ha behov av att göra en mer specialicerad bokning och man sätter då update till 2.

Med parametern update satt till 2 kan funktionen användas i stället för VoucherClose för att spara ett verifikat i samma transaktion som reskontraposten samtidigt som alla kopplingar uppdateras automatiskt (se exempel). I detta fall hanteras också verifikattexten på verifikat-raderna något annorlunda. Om man låter parametern voucherText vara blank i funktionen VoucherAddRow kommer Visma Compacts standardtext för leverantörsbokning att användas. Lägger man till egen text så kommer reskontratyp och löpnummer att infogas framför den egna texten.

Exempel

1) Lägg till en inscannad faktura till en leverantörsreskontrapost. Notera att fakturadokumentet måste läggas till innan reskontraposten skapas. När reskontraposten skapas kommer fakturadokumentet automatiskt att sparas i företagets databas och kopplas till reskontra och redovisning.

VB.Net

```
If DatabaseOpen("TESTBOLAG") = DbOk Then
      DocumentAddSupplierInvoice("C:¥Scanner¥faktura.pdf", True)
      LedgerAddSupplierInvoice( "10419", "123458", 2009, 6, 21, ↩
                1480.00, 296.00, FA, 1, "")
End If
DatabaseClose()
```

2) Boka ett verifikat för en leverantörsreskontrapost och låt funktionen `LedgerAddSupplierInvoice` avsluta verifikatet och spara det i samma transaktion som leverantörsreskontraposten sparas. Kopplingar mellan posterna uppdateras automatiskt. Observera att parametern update måste vara satt till 2 och LedgerAddSupplierInvoice måste anropas efter att alla rader lagts till. Reskontratyp och löpnummer kommer automatiskt att infogas innan verifikationstexten.

VB.Net

```
Dim amount As Double = 2500.0
Dim vat As Double = 500.0

If DatabaseOpen("TESTBOLAG") = DbOk Then

  If VoucherOpen(3, 2009, 10, 21) Then

    VoucherAddRow("2440", "Linneas Maskinimport", "", "", -amount)
    VoucherAddRow("2640", "Linneas Maskinimport", "", "", vat)
    VoucherAddRow("5400", "Linneas Maskinimport", "", "", amount-vat)

    LedgerAddSupplierInvoice("1041219", "123459", 2009, 10, 21, ↩
       amount, vat, FA, 2, "")

  End If
End If
```

DocumentAddSupplierInvoice

C

```
void WINAPI DocumentAddSupplierInvoice( LPCSTR filePath,
int storeDocument);
```

Visual Basic.Net

```
Declare Ansi Function GetStandardText Lib "XORBASE5.DLL"
(ByVal TextNumber As Integer, ByVal TextString As StringBuilder,
ByVal StringSize As Integer) As Integer
```

Används tillsammans med funktionen LedgerAddSupplierInvoice för att lägga till ett fakturadokument till en leverantörsreskontrapost. Om användaren har valt att inte spara dokument kommer enbart kopplingen till filnamnet (utan sökväg) att göras oberoende av värdet på storeDocument. Se också funktionen LedgerAddSupplierInvoice.

Parametrar

filePath	En komplett sökväg till dokumentet
storeDocument	Sätts till 1 om dokumentet skall sparas i företagets databas. Sätts till 0 om enbart kopplingen till dokumentets namn skall göras.

Returnerar
Inget

Anmärkning
Stöd för kopplade dokument finns i version 5.1.4 och senare av Visma Compact.

Exempel
Lägg till en inscannad faktura till en leverantörsreskontrapost. Notera att fakturadokumentet måste läggas till innan reskontraposten skapas. När reskontraposten skapas kommer fakturadokumentet automatiskt att sparas i företagets databas och kopplas till reskontra och redovisning.

VB.Net

```
If DatabaseOpen("TESTBOLAG") = DbOk Then
    DocumentAddSupplierInvoice("C:¥Scanner¥faktura.pdf", True)
    LedgerAddSupplierInvoice( "10419", "123458", 2009, 6, 21,
            1480.00, 296.00, FA, 1, "")
End If
DatabaseClose()
```

GetReservedAccount

```
C
      int WINAPI GetReservedAccount( int defaultNumber, ↩
      LPSTR accountNumber, int size);

Visual Basic.Net
      Declare Ansi Function GetReservedAccount Lib "XORBASE5.DLL" ↩
      (ByVal DefaultNumber As Integer, ↩
      ByVal AccountNumber As StringBuilder, ↩
      ByVal StringSize As Integer) As Integer
```

Hämtar ett reserverat konto från databasens register för reserverade konton.

Parametrar

defaultNumber	Ett fördefinierat värde som identiferar kontot (se nedan)
accountNumber	En buffert som tar emot kontonumret
size	Storleken på bufferten

Returnerar

DbOk	Kontot hittades
-1	Databasen var inte öppen
-2	Kontot kunde inte hittas

Anmärkning

De fördefinierade kontona som används av Visma Compact är:

DEFVATOUT	Utgående moms
DEFVATIN	Ingående moms
DEFSTOCK	Lagerkonto
DEFYEARRESULT	Årets resultat, balans
DEFCUSTOMER	Kundfordringar
DEFFREIGHT	Frakt
DEFFEE	Expeditionsavgift
DEFADJUSTMENT	Öresavrundning
DEFINTERIN	Ränteintäkter
DEFFINANCE	Finansiella poster
DEFSALES	Försäljning
DEFPURCHASE	Inköp
DEFSUPPLIER	Leverantörskuld
DEFPAYIN	Inbetalningskonto (bank)
DEFCASH	Kassakonto
DEFINTEROUT	Ränteutgifter
DEFCHANGE	Justeringskonto (lager)
DEFDISCOUNTOUT	Utgående kassarabatt
DEFDISCOUNTIN	Ingående kassarabatt
DEFXPROFIT	Kursvinst

DEFXLOSS	Kursförlust
DEFEXPSALES	Exportförsäljning EU
DEFEXPSALES2	Exportförsäljning
DEFPAYOUT	Utbetalningskonto (bank)
DEFVAT	Momsavräkning
DEFFREIGHTEXP	Frakt Export
DEFFEEEXP	Expeditionsavgift Export
DEFYEARRESULTBOOK	Årets resultat, resultat
DEFFFREIGHTEU	Expeditionsavgift Export EU
DEFFEEEU	Frakt Export EU
DEFREVVAT	Konto för omvänd momsplikt (byggmoms)

GetStandardText

```
C
	int WINAPI GetStandardText( int textNumber, LPSTR buffer, ↩
	int bufferSize);

Visual Basic.Net
	Declare Ansi Function GetStandardText Lib "XORBASE5.DLL" ↩
	(ByVal TextNumber As Integer, ByVal TextString As StringBuilder, ↩
	ByVal StringSize As Integer) As Integer
```

Hämtar en fördefinierad standardtext från databasens textregister.

Parametrar

textNumber	Textens ID-nummer
buffer	En buffert som tar emot texten
bufferSize	Storleken på bufferten

Returnerar

DbOk	Texten hämtades utan fel
-1	Databasen är inte öppen
-2	Texten kunde inte hittas

PersonFind

```
C
      int WINAPI PersonFind(int personType, LPCSTR personID);

Visual Basic.Net
      Declare Ansi Function PersonFind Lib "XORBASE5.DLL" ⮐
      (ByVal personType As Integer, ByVal personID As String) As Integer
```

Söker upp en existerande kund eller leverantör och sätter den som aktuell post.

Parametrar

personType	Persontyp: CUSTOM = kund SUPPLY = leverantör
personID	Person-ID

Returnerar

0	Ingen post hittades
1	Posten hittades

Exempel

Leta upp en kund och uppdatera referenspersonens namn.

VB.Net

```
If DatabaseOpen("TESTBOLAG") = DbOk Then
    Dim pRef As IntPtr = RecordField(PERSON, "REF")
    If pRef <> 0 Then
        If PersonFind(CUSTOM, "1005") Then
            FieldAssignString(pRef, "Kalle Karlsson")
            RecordUpdate(PERSON)
        End If

    End If
End If
DatabaseClose()
```

PersonAdd

C

```
int WINAPI PersonAdd( int personType, LPCSTR personID, ↩
LPCSTR personName, LPCSTR group, LPCSTR adress1, LPCSTR adress2, ↩
LPCSTR adress3, LPCSTR adress4, LPCSTR phoneNumber, ↩
LPCSTR faxNumber, LPCSTR reference, ↩
int termsOfPayment, int termsOfDelivery, int wayOfDelivery);
```

Visual Basic.Net

```
Declare Ansi Function PersonAdd Lib "XORBASE5.DLL" ↩
( ByVal PersonType As Integer, ByVal PersonID As String, ↩
ByVal PersonName As String, ByVal Group As String, ↩
ByVal Address1 As String, ByVal Address2 As String, ↩
ByVal Address3 As String, ByVal Address4 As String, ↩
ByVal PhoneNumber As String, ByVal FaxNumber As String, ↩
ByVal Reference As String, ↩
Optional ByVal TermsOfPayment As Integer = -1, ↩
Optional ByVal TermsOfDelivery As Integer = -1, ↩
Optional ByVal WayOfDelivery As Integer = -1) As Integerr
```

Lägger till en ny kund eller leverantör till databasen.

Parametrar

personType	Persontyp: CUSTOM = kund SUPPLY = leverantör
personID	Person-ID
personName	Personens namn
group	Persongrupp om sådan finnes
adress1	Besöksadress
adress2	Boxadress
adress3	Postnummer och ort
adress4	Land
phoneNumber	Telefonnummer
faxNumber	Telefaxnummer
reference	Referens
termsOfPayment	Betalningsvillkor, ange -1 för standardvärde
termsOfDelivery	Leveransvillkor, ange -1 för standardvärde
wayOfDelivery	Leveranssätt, ange -1 för standardvärde

Returnerar

DbOk	Personen lades till utan fel
-1	Personen finns redan
-2	Person-ID är för långt

Anmärkning

Om något textfält är för långt kommer det att klippas av. Standardvärden för betalningsvillkor, leveransvillkor och leveranssätt hämtas från företagsinställningarna. Standardvärden för de konton som förknippas med personen hämtas från reserverade konton.

Exempel

Lägg till en ny kund och lägg till en webbadress genom att uppdatera den nyss skapade posten.

C++

```
void AddPerson()
{
    Field web = RecordField( PERSON, "WEB");

    if( PersonAdd( CUSTOM, "12345", "Alfred Nyman", "", "Storgatan 12",
                   "Box 123", "222 44 Köpingsbruk", "", "048-112233",
                    "048-112232", "Kurt", -1, -1, -1) > -1 )
    {
        FieldAssignString( web, "www.alfrednyman.se");
        RecordUpdate(PERSON);
    }
}
```

ArticleFind

```
C
      int WINAPI ArticleFind( LPCSTR artNo);

Visual Basic.Net
      Declare Ansi Function ArticleFind Lib "XORBASE5.DLL" ⮠
      (ByVal artNo As String) As Integer
```

Söker upp en existerande artikel och sätter den som aktuell post.

Parametrar

artNo	Artikelns ID-nummer

Returnerar

0	Ingen post hittades
1	Posten hittades

ArticleAdd

```
C
        int WINAPI ArticleAdd( LPCSTR artNo, LPCSTR name, LPCSTR group, ↩
        double itemsStock, double minItems, double price, ↩
        double purchasePrice);

Visual Basic.Net
        Declare Ansi Function ArticleAdd Lib "XORBASE5.DLL" ↩
        (ByVal ArticleNumber As String, ByVal ArticleName As String, ↩
        ByVal Group As String, ByVal ItemsInStock As Double, ↩
        ByVal OrderPoint As Double, ByVal Price As Double, ↩
        ByVal PurchasePrice As Double) As Integer
```

Lägger till en ny artikel i databasen.

Parametrar

artNo	Den nya artikelns ID-nummer
name	Namnet på den nya artikeln
group	Artikelns grupp
itemsStock	Antal i lager
minItems	Beställningspunkt
price	Artikelns försäljningspris
purchasePrice	Artikelns inköpspris

Returnerar

DbOk	Artikeln lades till utan fel
-1	Artikeln finns redan i databasen
-2	Artikelns ID är för långt

Anmärkning

Artikelgrupp är ett frivilligt fält. Om artikelns grupp eller namn är längre än databasfältet kommer de att klippas av. I samband med skapandet av artikeln sätts alla konton förknippade med artikeln till värden från reserverade konton.

ObjectFind

```
C
      int WINAPI ObjectFind( LPCSTR objectID);

Visual Basic.Net
      Declare Ansi Function ArticleFind Lib "XORBASE5.DLL" ⮠
      (ByVal objectID As String) As Integer
```

Söker upp ett existerande objekt och sätter det som aktuell post.

Parametrar

objectID	Objektets id

Returnerar

0	Ingen post hittades
1	Posten hittades

ObjectAdd

```
C
      int WINAPI ObjectAdd( int objectType, LPCSTR objectID, ↩
      LPCSTR name, int spanYear);

Visual Basic.Net
      Declare Ansi Function ObjectAdd Lib "XORBASE5.DLL" ↩
      ( ByVal objectType As Integer, ByVal ObjectID As String, ↩
      ByVal Name As String, ByVal SpanYear As Integer) As Integer
```

Lägger till ett nytt objekt i databasen.

Parametrar

objectType	Typ av objekt, 1 eller 2
objectID	Det nya objektets id
spanYear	0 för ettårigt objekt, 1 för flerårigt

Returnerar

DbOk	Objektet har lagts till utan fel
-1	Objektet finns redan
-2	ID-strängen för lång

Anmärkning

Om objektets namn är för långt för namnfältet kommer det att klippas av.

ClientListInit

```
C
      int WINAPI ClientListInit(void * clientName);

Visual Basic.Net
      Declare Ansi Function ClientListInit Lib "XORBASE5.DLL" ↩
      (<[In](), Out()> ByRef clientName As ClientNameType) As Integer
```

Initierar en iterator för att kunna iterera igenom Visma Compacts företagslista. Används tillsammans med ClientListNext för att kunna hämta företagsnummer för alla tillgängliga företag.

Parametrar

clientName	En datastruktur som innehåller företagsid och namn

Returnerar

<> 0	Om clientName innehåller ett företag

Anmärkning

Exempel

Iterera igenom företagslistan och fyll en listvy med data.

```
VB.Net
        Dim result As Boolean
        Dim client As New ClientNameType
        Dim item As ListViewItem

        result = ClientListInit(client) > 0

        While result
            item = New ListViewItem(client.id)
            item.Name = client.id
            item.SubItems.Add(client.name)
            lvClients.Items.Add(item)
            result = ClientListNext(client) > 0
```

ClientListNext

C

```
int WINAPI ClientListNext(void * clientName);
```

Visual Basic.Net

```
Declare Ansi Function ClientListNext Lib "XORBASE5.DLL" ↵
(<[In](), Out()> ByRef clientName As ClientNameType) As Integer
```

Hämta nästa företag i Visma Compacts företagslista. Används tillsammans med ClientListInit för att kunna hämta företagsnummer för alla tillgängliga företag.

Parametrar

clientName	En datastruktur som innehåller företagsid och namn

Returnerar

<> 0	Om clientName innehåller ett företag

Anmärkning

ExportSieFile

C

```
int WINAPI ExportSieFile( LPCSTR fileName, int level, ↩
WORD julianDate);
```

Visual Basic.Net

```
Declare Ansi Function ExportSieFile Lib "XORBASE5.DLL" ↩
(ByVal FileName As String, Optional ByVal Level As Integer = 4, ↩
Optional ByVal julianDate As Short = 0) As Integer
```

Exporterar redovisningsdata i SIE-format till en namngiven fil.

Parametrar

fileName	Ett filnamn inklusive filändelse
level	Nivå på exporten (1-4)
julianDate	Datum för det räkenskapsår som skall exporteras. Ange 0 för innevarande år.

Returnerar

-1	Databasen kan inte öppnas
-2	Felaktig SIE nivå
-3	Räkenskapsår saknas för angivet datum
-4	Kan inte skapa fil
-9	Organisationsnummer saknas i företagsinställningarna

Anmärkning
Se också www.sie.se

Exempel

Exportera en SIE-fil med redovisningsdata för innevarande år enligt SIE nivå 1.

VB.Net

```
If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
       If DatabaseOpen(clientId) = DbOk Then
              Dim ret As Integer
            ret = ExportSieFile(SaveFileDialog1.FileName(), 1, 0)
            DatabaseClose()
       End If
End If
```

ImportSieFile

```
C
      int WINAPI ImportSieFile( LPCSTR fileName);

Visual Basic.Net
      Declare Ansi Function ImportSieFile Lib "XORBASE5.DLL" ↩
      (ByVal FileName As String) As Integer
```

Importerar redovisningsdata i SIE-format från en namngiven fil.

Parametrar

fileName	Ett filnamn

Returnerar

>= 0	Antal verifikat som importerades
-1	Databasen kan inte öppnas
-3	Filen har redan importerats
-4	Felaktigt format
-5	Felaktig SIE typ
-8	Felaktig kontrollsumma
-9	Organisationsnumret i SIE-filen överensstämmer inte med företagets
-10	Fel vid import
-11	Kan inte öppna filen
-13	Räkenskapsåret överensstämmer inte med företagets
-17	Bokföringsdatum utanför räkenskapsår

Anmärkning

Enbart filer av SIE nivå 4 kan importeras till Visma Compact. Se också www.sie.se.

DueDateCalc

```
C
        WORD WINAPI DueDateCalc( LPCSTR termsOfPayment, WORD startDate);

Visual Basic.Net
        Declare Ansi Function DueDateCalc Lib "XORBASE5.DLL" ⮐
        (ByVal TermsOfPayment As String, ByVal StartDate As Short) As Short
```

Beräkna förfallodag givet startdag och en textsträng med betalningsvillkor.

Parametrar

termsOfPayment	Textsträng innehållande betalningsvillkor
startDate	Julianskt datum från vilket förfallodagen skall beräknas

Returnerar

> 0	Ett julianskt datum

Anmärkning

Funktionen använder en enkel algoritm för att beräkna förfallodatum baserat på vanligt förekommande betalningsvillkor. T ex *10 dagar netto* eller *Fri månad + 15 dagar*.

Villkor:

Inga siffror i strängen: 0 dagar
Siffror i början av strängen: NN dagar
Siffror inuti strängen: fri månad +/- NN dagar

DateSystem

```
C
        WORD WINAPI DateSystem(void);

Visual Basic.Net
        Declare Ansi Function DateSystem Lib "XORBASE5.DLL" () As Short
```

Returnerar ett julianskt datum baserat på lokal tid från datorns interna systemklocka.

Parametrar

(inga)

Returnerar

> 0	Ett julianskt datum

YMDToDate

```
C
      WORD WINAPI YMDToDate(int year, int mon, int day, int check);

Visual Basic.Net
      Declare Ansi Function YMDToDate Lib "XORBASE5.DLL" ⮐
      (ByVal year As Integer, ByVal mon As Integer, ⮐
      ByVal day As Integer, ByVal check As Integer) As Short
```

Konverterar ett datum angivit som år, månad, dag till ett julianskt datum.

Parametrar

year	År, angivet med fyra siffror
mon	Månad (1 – 12)
day	Dag (1 – 31)
check	Validera invärden

Returnerar

0	Om check = 1 och datumet är ogiltligt
> 0	Ett julianskt datum

Anmärkning
Om check = 0 och datumet är ogiltligt returneras ett julianskt datum baserat på en uppskattning av invärdena.

FirstDayOfWeek

```
C
      WORD WINAPI FirstDayOfWeek( int year);

Visual Basic.Net
      Declare Ansi Function FirstDayOfWeek Lib "XORBASE5.DLL" ↵
      (ByVal year As Integer) As Short
```

Returnerar veckodagen för årets första dag.

Parametrar

year	Året, angivet med fyra siffror

Returnerar

1 – 7	Veckodagen, där 1 = måndag

DayOfWeek

```
C
      WORD WINAPI DayOfWeek( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function DayOfWeek Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar veckodagen för givet julianskt datum.

Parametrar

julianDate	Julianskt datum

Returnerar

1 – 7	Veckodagen, där 1 = måndag

BeginWeek

```
C
      WORD WINAPI BeginWeek( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function BeginWeek Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för första dagen i veckan.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

EndWeek

```
C
      WORD WINAPI EndWeek( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function EndWeek Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för sista dagen i veckan.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

EndWorkWeek

```
C
        WORD WINAPI EndWorkWeek( WORD julianDate );

Visual Basic.Net
        Declare Ansi Function EndWorkWeek Lib "XORBASE5.DLL" ⮠
        (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för sista arbetsdagen i veckan.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

DayOfYear

```
C
      WORD WINAPI DayOfYear( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function DayOfYear Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar dagens nummer i året.

Parametrar

julianDate	Julianskt datum

Returnerar

1 – 366	Dagens nummer

BeginYear

```
C
      WORD WINAPI BeginYear( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function BeginYear Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för första dagen i året.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

EndYear

```
C
      WORD WINAPI EndYear( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function EndYear Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för sista dagen i året.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

BeginMon

```
C
      WORD WINAPI BeginMon( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function BeginMon Lib "XORBASE5.DLL" ⮐
      (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för första dagen i månaden.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

EndMon

```
C
      WORD WINAPI EndMon( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function EndMon Lib "XORBASE5.DLL" ↩
      (ByVal julianDate As Short) As Short
```

Beräknar det julianska datumet för sista dagen i månaden.

Parametrar

julianDate	Julianskt datum

Returnerar

> 0	Julianskt datum

Year

```
C
      int WINAPI Year( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function Year Lib "XORBASE5.DLL" ↩
      (ByVal julianDate As Short) As Integer
```

Beräknar året från givet julianskt datum.

Parametrar

julianDate	Julianskt datum

Returnerar

> 1980	Året (ex. 2006)

Month

```
C
      int WINAPI Month( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function Month Lib "XORBASE5.DLL" ↩
      (ByVal julianDate As Short) As Integer
```

Beräknar månadens nummer från givet julianskt datum.

Parametrar

julianDate	Julianskt datum

Returnerar

1 – 12	Månadens nummer

Day

```
C
      int WINAPI Day( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function Day Lib "XORBASE5.DLL" ↩
      (ByVal julianDate As Short) As Integer
```

Beräknar dagens nummer från givet julianskt datum.

Parametrar

julianDate	Julianskt datum

Returnerar

1 – 31	Dagens nummer i månaden

Week

```
C
      int WINAPI Week( WORD julianDate );

Visual Basic.Net
      Declare Ansi Function Week Lib "XORBASE5.DLL" ↩
      (ByVal julianDate As Short) As Integer
```

Beräknar årets veckonummer från givet julianskt datum.

Parametrar

julianDate	Julianskt datum

Returnerar

1 – 53	Årets veckonummer

GetYearNo

C

```
int WINAPI GetYearNo( WORD julianDate);
```

Visual Basic.Net

```
Declare Ansi Function GetYearNo Lib "XORBASE5.DLL" ↩
(ByVal julianDate As Short) As Integer
```

Hämtar räkenskapsårets id för ett givet julianskt datum. Om julianDate sätts till 0 används dagens datum.

Parametrar

julianDate	Julianskt datum

Returnerar

-1	Databasen är inte öppen
-2	Det finns inget räkenskapsår upplagt för detta datum

IsDateLocked

C

```
int WINAPI IsDateLocked( WORD julianDate );
```

Visual Basic.Net

```
Declare Ansi Function IsDateLocked Lib "XORBASE5.DLL" ↩
(ByVal julianDate As Short) As Integer
```

Kontrollerar om ett bokföringsdatum infaller under ett stängt år eller en låst period.

Parametrar

julianDate	Julianskt datum

Returnerar

1	Datum infaller under ett stängt år eller en låst period
0	Datum infaller i en öppen bokföringsperiod
-1	Databasen är inte öppen
-2	Inget räkenskapsår för detta datum