
# Dokumentation
# Xpert-Timer API (XTAPI.DLL)

**Version:** 6.0

**DB-Patchlevel:** PL 88

**DLL-Revision:** R7

**WICHTIG: Bitte beachten Sie die Änderungshistorie am Ende dieses Dokuments falls Sie von R3 auf R4,R5,R6,R7 updaten.**

# Inhaltsverzeichnis

# 1 Allgemeines

Die XTAPI ist eine Schnittstelle um datenbankunabhängig auf die Datensätze des Programms zugreifen zu können.

Die aktuelle Version dieser Dokumentation finden Sie unter:
http://download.xpertdesign.de/helpfiles/XTAPI_Doc.pdf

Die XTAPI.DLL inkl. Demo Sourcecode als ZIP finden Sie unter:
http://download.xpertdesign.de/additional/XTAPIDemo.zip

Folgende Möglichkeiten sind vorgesehen:

- Datenbankverbindung aufbauen / trennen
- Benutzer einloggen/ausloggen
- Projekt, Kunden, Benutzer, Aufgaben, Historieneinträge, Zeitstempel anlegen, ändern oder löschen.
- Vorhandene Projekte, Kunden, Benutzer, Aufgaben, Historien, Zeitstempellisten auslesen
- Zeitstempel oder Historieneinträge nachtragen
- Daten über den angemeldeten Benutzer abfragen
- Systemdaten abfragen
- Projekte zu Benutzern zuweisen
- Projekt- Kunden- Benutzer, Aufgaben, Historien, Zeitstempeldetails auslesen und ändern
- Projekte starten, stoppen oder pausieren (Fernsteuerung einer laufenden XT-Instanz)
- Berechnen von Zeitstempelsummen unter Berücksichtigung von Benutzer, Projekt, Kunde, Zeitraum
- Drucken beliebiger mit XTReport erstellte Reportdateien (.fr3)
- Auswahldialoge öffnen für Benutzer, Kunden, Projekte
- Bearbeitungsdialoge öffnen für Projekt, Kunden, Benutzer, Aufgaben, Historieneinträge, Zeitstempel
- Hilfsfunktionen zur Datumsverarbeitung
- Zeitstempelliste drucken/exportieren

# 2 Anbinden über Programmiersprachen

## 2.1 Allgemeines

Damit die XTAPI.DLL gefunden werden kann, sollte sich diese im System-Verzeichnis der Windowsinstallation befinden (C:\WINDOWS\system32). Dort steht sie allen Programmen zur Verfügung. Die DLL kann jedoch auch in das Verzeichnis kopiert werden, in dem sich das aufrufende Programm befindet (.EXE Datei), um sicherzustellen, daß das Programm immer genau eine bestimmte Version der DLL verwendet. Dieses Vorgehen kann wesentliche Vorteile und schwere Nachteile zur Folge haben.

## 2.2 Delphi

Die Anbindung an Delphi erfolgt durch die Unit „XTAPIDLLHeader.pas“ , die im Quellcode vorliegt. Diese Unit wird in einem Programm verwendet durch das Schlüsselwort **uses** (siehe Dokumentation zu Delphi).

Siehe Beispiel „Examples\Delphi“

## 2.3 VB.NET

Die Anbindung an Microsoft VisualBasic erfolgt durch das Modul „XTAPIVBNET.VB“, das die benötigten Deklarationen enthält. Diese Datei wird einem VB.NET-Projekt hinzugefügt. Danach stehen die Deklarationen und damit auch die Funktionen global zur Verfügung.

Siehe Beispiel „Examples\VB.NET“

## 2.4 C und C++

Die Anbindung an die Programmiersprachen C und C++ erfolgt mit der Header-Datei XTAPI.H sowie einer Link-Library, die zum Linken als Bibliothek mit angegeben wird und die Verbindung zur DLL herstellt. Link-Libraries für viele Compiler und Entwicklungsumgebungen können mit Hilfe dazugehöriger HIlfsprogramme direkt aus der DLL erstellt werden. Die Anwendung kann dem Demoprogramm entnommen werden.

Siehe Beispiel „Examples\C++“

## 2.5 Excel (VBA)

Die Anbindung an Microsoft Excel erfolgt durch das Modul „XTAPIVBA.BAS“, das die benötigten Deklarationen enthält. Diese Datei wird einem Excel-Projekt hinzugefügt. Danach stehen die Deklarationen und damit auch die Funktionen global zur Verfügung.

Siehe Beispiel „Examples\Excel“

## 2.6 Visual Basic

Die Anbindung an Microsoft VisualBasic erfolgt durch das Modul „XTAPIVB.BAS“, das die benötigten Deklarationen enthält. Diese Datei wird einem VisualBasic-Projekt hinzugefügt. Danach stehen die Deklarationen und damit auch die Funktionen global zur Verfügung.

Siehe Beispiel „Examples\VB“

## 2.7 C#

Siehe Beispiel „Examples\C#“

# 3 Datenobjekte

Folgende Stammdatenobjekte bilden den Kern des Xpert-Timers.

- Benutzer
- Kunden / Kundenadressen
- Projekte

Für diese Stammdatenobjekte können folgende Erfassungsdatenobjekte erstellt werden:

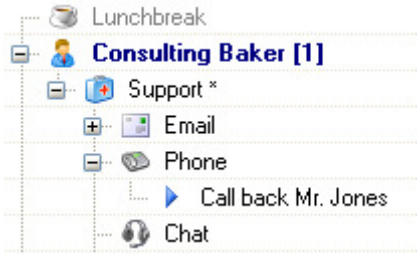
- Aufgaben
- Historieneinträge
- Zeitstempel

Der hierarchische Aufbau ist immer wie folgt:

[Kunde]
Projekt
[Unterprojekt]
Benutzer
Aufgabe
Historie
Zeitstempel

Objekte in eckigen Klammern „[]“ sind hierbei optional. D.h. ein Projekt muss nicht an einem Kunden hängen und eine Unterprojektebene ist nicht verpflichtend. Es ist jedoch auch nur maximal eine Unterprojektebene möglich.

z.B.



## 3.1 Allgemeine Verfahrensweisen

### 3.1.1 Primärschlüssel (GUID)

Datenobjekte haben im Xpert-Timer immer eine eindeutige Kennung über welche sie angesprochen werden. Eine sog. GUID (Global Unique Identifier). Diese GUID ist eine 38-Stellige alphanumerische Kennung die es ermöglicht sehr komfortabel ohne Nummerkreise zu arbeiten.

Beispiel:
`{0767F4E9-CC24-4D6A-87DA-F6D9B088D503}`

Eine GUID kann über die Funktion „XTCreateGUID“ erstellt werden.

Falls ein Datensatz eine solche GUID enthält, so ist das entsprechende Feld immer durch die Endung „`_nr`“ gekennzeichnet. Z.B. `user_nr`, `client_nr`, `project_nr` usw.

Es gibt im Xpert-Timer zwei verschiedene Sonderfälle einer GUID um den Zustand NULL bzw. „nicht verknüpft“ zu kennzeichnen. Diese werden benötigt um die referenzielle Integrität für SQL-JOINS zur erhalten und vereinfacht somit das auslesen von SQL-Datenmenge erheblich.

```
{11111111-1111-1111-1111-111111111111} = Verknüpfung mit leerem Datenobjekt
{00000000-0000-0000-0000-000000000000} = Leerer Feldinhalt
```

### 3.1.2 Booleanwerte True/False

Boolwerte werden über die Konstanten XTC_TRUE (=1) und XTC_FALSE (=0) abgebildet.

### 3.1.3 Löschen von Datensätzen

Datensätze werden innerhalb der XT-Datenbank niemals wirklich gelöscht, sondern nur als „gelöscht“ gekennzeichnet.

### 3.1.4 Datenbank

Der Xpert-Timer unterstützt folgende Datenbanken:

- Microsoft Access
- Microsoft SQL Server
- MySQL
- SQLite
- PostgreSQL

Die Konfiguration der Datenbankverbindung geschieht optimalerweise über das mitgelieferte Programm XTADMIN.exe. Die damit vorgenommenen Einstellungen werden dann in der Datei `xperttimer.ini` gespeichert. Durch das kopieren dieser Datei zur XTAPI.DLL kann ohne weitere Konfiguration die Datenbankverbindung automatisch hergestellt werden. Alternativ dazu besteht auch die Möglichkeit die Verbindungsdaten (z.B. zum MS-SQL-Server) über eine DLL-Funktion zu konfigurieren.

### 3.1.5 Hinweise zum Login

Generell können sich über die XTAPI nur Benutzer mit Administratorenrechten anmelden. Es gibt jedoch einen Ini-Schalter in der "xperttimer.ini" um von "Nur Admin" auf "Allgemeine Anmeldung" bei der DLL-Benutzung umzuschalten.

```
[XTAPI]
   AdminOnly=0
```

Desweiteren gibt es die Möglichkeit einzelnen Benutzern das verwenden der XTAPI zu verbieten/erlauben. Dafür gibt es in den Mitarbeitereigenschaften das Feld „XTAPI-Zugriff“

### 3.1.6 Kommentare und Beschreibungsfelder

Technisch bedingt können bei Verwendung der Objektstrukturen innerhalb der XTAPI.DLL keine längeren Texte als max. 255 Zeichen verarbeitet werden. I.d.R. gibt es für jedes längere Textfeld ein entsprechendes Flag in der Struktur welches angibt ob der in der Datenbank gespeicherte Text länger ist, als der der eingelesen wurde. Z.B. „project.Info“ und „project.info_overflow“. Das entsprechende Textfeld wird bei gesetztem Overflow-Flag dann nicht in die Datenbank zurückgespeichert um den Originaltext so nicht unwissentlich zu kürzen.

### 3.1.7 Fehlersuche

Jede Aufgerufene DLL-Funktion gibt seine Aufrufparameter über OutputDebugString im Windows-Debug-Log aus.  Dieses kann über Programme wie z.B. DBGVIEW.EXE (erhältlich unter www.sysinternals.com) ausgelesen werden.

### 3.1.8 Suchpfade der DLL

Die XTAPI.DLL kann entweder im Programmverzeichnis des aufrufenden Programms liegen, oder direkt im Windows-Systemverzeichnis (C:\Windows\system32, bzw. c:\windows\SysWOW64). Für Programme die die DLL statisch linken (z.B. VBA unter Excel), ist es sinnvoll die DLL im Windows-Systemverz. liegen zu haben.

## 3.2 Benutzer

Ein Benutzer ist ein Mitarbeiter in der Datenbank. Er wird i.d.R. mit seiner eindeutigen `user_nr` referenziert. Der Ersteller eines Datensatzes wird i.d.R. über `creator_nr` angesprochen.

```
typedef struct _XTUserdata {
    char szUser_nr[GUID_LENGTH]; // Eindeutige GUID                    [Pflichtfeld]
    char szLoginname[50];          // Loginname für Benutzeranmeldung   [Pflichtfeld]
    char szFirstname[50];        // Vorname                           [Pflichtfeld]
    char szLastname[50];         // Nachname                          [Pflichtfeld]
    char szEmail[128];           // Emailadresse
    char szCostcenter[20];       // Kostenstelle
    short bChangePwOnLogin;      // Passwort ändern bei Erstem Login?
    short bActive;               // Ist der Benutzer Aktiviert?
    short bIsAdmin;              // Ist der Benutzer Administrator?
    short bIsLoggedIn;           // Ist der Benutzer Eingeloggt?
    int iPersnr;                 // Personalnummer
    int iSecuritylevel;          // Sicherheitsstufe
    int fPricePerBillingUnitInternal; // Preis pro Einheit intern
    int fPricePerBillingUnit;    // Preis pro Einheit
    int iBillingUnit;            // Abrechnungseinheit
    SYSTEMTIME dtLastLogin;                 // Letzter Login
    SYSTEMTIME dtLastLogout;                // Letzter Logout
    char szLoginComputer[80];               // Name des Logincomputers
    char szActiveproject_nr[GUID_LENGTH]; // GUID des laufenden Projekts
    char szActivetodo_nr[GUID_LENGTH];    // GUID der laufenden Aufgabe
    char szCreator_nr[GUID_LENGTH];       // GUID des Erstellers
    SYSTEMTIME dtDatecreated;               // Erstellungsdatum
    SYSTEMTIME dtLastChange;                // Datum/Zeit der letzten Änderung

    char szMainworkgroup_nr[GUID_LENGTH]; // R7: GUID der Hauptarbeitsgruppe
    char szPricegroup_nr[GUID_LENGTH];    // R7: GUID der Standardpreisgruppe

    char szReserved[256];                   // reserviert für spätere Erweiterungen
} XTUserdata, *PXTUserdata, *LPXTUserdata;
```

## 3.3 Kunden

Die Kundenebene ist für die Zeiterfassung nicht erforderlich, erleichtert jedoch die Auftragsverwaltung erheblich.

```
typedef struct _XTClientdata {
     char szClient_nr[GUID_LENGTH];       // Eindeutige GUID              [Pflichtfeld]
     char szClientname[80];               // Kundename                    [Pflichtfeld]
     char szVatid[11];                    // UmsatzsteuerID
     char szInfo[255];                    // Kommentar zum Kunden
     short bInfo_overflow                 // Infotext > 255 Zeichen
     char szClientreference[20];          // Referenznummer
     int iClientid;                       // Kundennummer                 [Pflichtfeld]
     short bActive;                       // Ist der Kunde aktiv?
     char szUserdefined1[80];             // Benutzerdefiniertes Feld 1
     char szUserdefined2[80];             // Benutzerdefiniertes Feld 2
     char szUserdefined3[80];             // Benutzerdefiniertes Feld 3
     char szUserdefined4[80];             // Benutzerdefiniertes Feld 4
     char szUserdefined5[80];             // Benutzerdefiniertes Feld 5
     char szUserdefined6[80];             // Benutzerdefiniertes Feld 6
     char szUserdefined7[80];             // Benutzerdefiniertes Feld 7
     char szUserdefined8[80];             // Benutzerdefiniertes Feld 8
     char szUserdefined9[80];             // Benutzerdefiniertes Feld 9
     char szUserdefined10[80];            // Benutzerdefiniertes Feld 10
     int iRelationstate;                  // Einschätzung zu Kundenbeziehung
     char szRelationcomment[255];         // Info zur Kundenbeziehung
     SYSTEMTIME dtRelationstatedate;      // Datum der Einschätzung
     double fPriceperbillingunit;         // Preis pro Einheit
```

```
     int iBillingunit;                    // Abrechnungseinheit (in Minuten)
     char szCreator_nr[GUID_LENGTH];      // GUID des Erstellers
     SYSTEMTIME dtDatecreated;            // Erstellungsdatum
     SYSTEMTIME dtLastChange;             // Datum/Zeit der letzten Änderung
      int color;                                 // R7: Farbe
      int currency_lcid;                         // R7: Kundenwährung
      char szMaincontact_nr[GUID_LENGTH];        // R7: Hauptkontakt
      char szCreated_workgroup_nr[GUID_LENGTH];  // R7: Hauptarbeitsgruppe
     char szPricetable_id_nr[GUID_LENGTH];      // R7: Preisliste

     char szReserved[256];                // reserviert für spätere Erweiterungen
} XTClientdata, *PXTClientdata, *LPXTClientdata;
```

## 3.4 Kundenadressen

Die Kundenadressen werden verwendet um verschiedene Ansprechpartner in einem Unternehmen verwalten zu können. Eine Kundenadresse gehört immer fest zu einem Kunden.

```
typedef struct _XTClientContactdata {
     char szContact_nr[GUID_LENGTH];      // Eindeutige GUID              [Pflichtfeld]
     char szClient_nr[GUID_LENGTH];       // GUID des Kunden              [Pflichtfeld]
     char szName[50];                     // Name                         [Pflichtfeld]
     char szDescription[50];              // Beschreibung des Kontaktes
     char szPhone1[40];                   // Telefon 1
     char szPhone2[40];                   // Telefon 2
     char szEmail1[40];                   // Email 1
     char szEmail2[40];                   // Email 2
     char szAddress1[40];                 // Adresse 1
     char szAddress2[40];                 // Adresse 2
     char szCity[40];                     // Stadt
     char szZip[10];                      // PLZ
     char szCountry[20];                  // Land
     char szState[40];                    // Bundesland
     char szWebsite[255];                 // Webseite
     int iSalutation;                     // Anrede
     int iContacttype;                    // Addresstyp
     short bActive;                       // Ist die Adresse aktiv?
     SYSTEMTIME dtBirthday;               // Geburtstag
     char szComment[255];                 // Kommentar zum KOntakt
     short bComment_overflow;             // > 255 Zeichen
     char szTextualaddress[255];          // Addresse im Textformat
     short bTextualaddress_overflow;      // > 255 Zeichen
     char szCreator_nr[GUID_LENGTH];      // GUID des Erstellers
     SYSTEMTIME dtDatecreated;            // Erstellungsdatum
     SYSTEMTIME dtLastChange;             // Datum/Zeit der letzten Änderung

     char szUserdefined1[80];             // R7: Benutzerdefiniertes Feld 1
     char szUserdefined2[80];             // R7: Benutzerdefiniertes Feld 2
     char szUserdefined3[80];             // R7: Benutzerdefiniertes Feld 3
     char szUserdefined4[80];             // R7: Benutzerdefiniertes Feld 4
     char szUserdefined5[80];             // R7: Benutzerdefiniertes Feld 5
     char szUserdefined6[80];             // R7: Benutzerdefiniertes Feld 6
     char szUserdefined7[80];             // R7: Benutzerdefiniertes Feld 7
     char szUserdefined8[80];             // R7: Benutzerdefiniertes Feld 8
     char szUserdefined9[80];             // R7: Benutzerdefiniertes Feld 9
     char szUserdefined10[80];            // R7: Benutzerdefiniertes Feld 10

     char szReserved[256];                // reserviert für spätere Erweiterungen
} XTClientContactdata, *PXTClientContactdata, *LPXTClientContactdata;
```

## 3.5 Projekte

Die Projekte sind der Kernbestandteil des Xpert-Timer und deshalb ist für nahezu alle Aktionen eine `project_nr` notwendig.

```
typedef struct _XTProjectdata {
     char szProject_nr[GUID_LENGTH];       // Eindeutige GUID            [Pflichtfeld]
     char szParent_project_nr[GUID_LENGTH];  // Eindeutige GUID des Hauptprojekts
     char szClient_nr[GUID_LENGTH];        // Eindeutige GUID des Kunden
     char szProjectname[80];               // Name des Projekts          [Pflichtfeld]
     char szProjecttype_nr[GUID_LENGTH];   // GUID des Projekttyps
     char szProjectnumber[20];             // Alphanummerische Projektnummer
     char szDescription[255];              // Projektbeschreibung
     short bDescription_overflow;          // Ist Description länger als 255 Zeichen
     char szInCharge_nr[GUID_LENGTH];      // GUID des Verwantwortlichen
     SYSTEMTIME dtDatebegin;               // Geplanter Projektbeginn
     SYSTEMTIME dtDateend;                 // Geplantes Projektende
     SYSTEMTIME dtDatestarted;             // Tatsächlicher Projektbeginn
     SYSTEMTIME dtDatefinished;            // Tatsächliches Projektende
     int iState;                           // Projektstatus: 0=psRunning, 1=psPaused,
                                              2=psCancled, 3=psFinished, 4=psInactive,
                                              5=psAccounted
     int iPriority;                        // Priorität
     int iProgress;                        // Projektfortschritt in %
     int iMinutesneeded;                   // Benötigte Zeit in Minuten. Diese Feld ist
                                              READ-ONLY und wird aus den erfassten
                                              Zeitstempeln berechnet
     int iMinutesestimated;                // Geschätzter Aufwand in Minuten
     int iProgressmode;                    // Fortschrittsabfrage
     int iProgressquery;                   // Fortschrittsabfragemodus
     short bHasTimeaccount;                // Zeitkonto vorhanden?
     short bTeamproject;                   // Kann Projekt im Team bearbeitet werden?
     int iAccountmode;                     // Abrechnungsmodus
     int iFulltimestampmode;               // Modus für Zeitstempel runden
     int iFulltimestampThreshold;          // Rundungseinheit
     int iGetPriceFrom;                    // Woher kommt der Preis?
     double fPricePerBillingUnit;          // Preis pro Einheit
     double fFlatratePrice;                // Pauschalpreis
     int iBillingunit;                     // Abrechnungseinheit in Minuten
     char szCreator_nr[GUID_LENGTH];       // GUID des Erstellers
     SYSTEMTIME dtDatecreated;             // Erstellungsdatum
     SYSTEMTIME dtLastChange;              // Datum/Zeit der letzten Änderung

     char szSecondincharge_nr[GUID_LENGTH];  // R7: GUID des 2.Verantwortlichen
     char szMaincontact_nr[GUID_LENGTH];     // R7: GUID des Hauptkontakts
     char szCreated_workgroup_nr[GUID_LENGTH]; // R7: GUID der Hauptarbeitsgruppe
     char szPricetable_id_nr[GUID_LENGTH];   // R7: GUID der Preisliste

     char szUserdefined1[80];              // R7: Benutzerdefiniertes Feld 1
     char szUserdefined2[80];              // R7: Benutzerdefiniertes Feld 2
     char szUserdefined3[80];              // R7: Benutzerdefiniertes Feld 3
     char szUserdefined4[80];              // R7: Benutzerdefiniertes Feld 4
     char szUserdefined5[80];              // R7: Benutzerdefiniertes Feld 5
     char szUserdefined6[80];              // R7: Benutzerdefiniertes Feld 6
     char szUserdefined7[80];              // R7: Benutzerdefiniertes Feld 7
     char szUserdefined8[80];              // R7: Benutzerdefiniertes Feld 8
     char szUserdefined9[80];              // R7: Benutzerdefiniertes Feld 9
     char szUserdefined10[80];             // R7: Benutzerdefiniertes Feld 10

     char szReserved[256];                 // reserviert für spätere Erweiterungen
} XTProjectdata, *PXTProjectdata, *LPXTProjectdata;
```

## 3.6 Aufgaben

Über die Aufgaben können den Benutzern spezifische Tätigkeiten zugeordnet werden, welche zu einem Stichtag erledigt werden sollten.

```
typedef struct _XTTaskdata {
     char szTodo_nr[GUID_LENGTH];            // Eindeutige GUID            [Pflichtfeld]
```

```
    char szSubject[255];                    // Betreff                    [Pflichtfeld]
    char szText[255];                       // Textkörper
    short bText_overflow;                   // Überlaufkennung wenn Text beim Einlesen
                                               mehr als 255 Zeichen enthält
    int iTodonumber;                        // Laufende Aufgabennummer (wird
                                               automatisch vergeben)
    int iTaskstate;                         // Aufgabenstatus: 0=xttdstNotStarted,
                                               1=xttdstChecking, 2=xttdstInProgress,
                                               3=xttdstPaused, 4=xttdstWaiting,
                                               5=xttdstDeclined, 6=xttdstDone
    int iProgress;                          // Fortschritt in %
    int iPriority;                          // 5 Stufige Priorität
                                               (0=Low;...;5=Highest)
    int iMinutesestimated;                  // Geschätzter Aufwand
    int iMinutesNeeded;                     // Benötigte Zeit
    SYSTEMTIME dtDatedue;                   // Fälligkeitsdatum
    SYSTEMTIME dtDatestart;                 // Aufgabenbeginn
    SYSTEMTIME dtDateread;                  // Gelesen am
    SYSTEMTIME dtDatedone;                  // Erledigt am
    char szTodocategory_nr[GUID_LENGTH];    // GUID der Kategorie
    char szProject_nr[GUID_LENGTH];         // GUID des Projekts
    char szUser_nr[GUID_LENGTH];            // GUID des Empfängers        [Pflichtfeld]
    char szCreator_nr[GUID_LENGTH];         // GUID des Erstellers        [Pflichtfeld]
    SYSTEMTIME dtDatecreated;             // Erstellungsdatum
    SYSTEMTIME dtLastChange;              // Datum/Zeit der letzten Änderung
    char szTaskstatecomment[200];          // Notiz zum Aufgabenstatus
    short bTaskstatecomment_overflow;     // Ist Notiz länger als 200 Zeichen

    char szUserdefined1[80];              // R7: Benutzerdefiniertes Feld 1
    char szUserdefined2[80];              // R7: Benutzerdefiniertes Feld 2
    char szUserdefined3[80];              // R7: Benutzerdefiniertes Feld 3
    char szUserdefined4[80];              // R7: Benutzerdefiniertes Feld 4
    char szUserdefined5[80];              // R7: Benutzerdefiniertes Feld 5

    char szReserved[53];                  // reserviert für spätere Erweiterungen
} XTTaskdata, *PXTTaskdata, *LPXTTaskdata;
```

## 3.7 Historieneintrag

Ein Historieneintrag enthält einen Erfassungsdatensatz, welcher den Fortschritt des Projekts beschreibt und wird i.d.R. als Projektlogbuch verwendet. Er ist immer verpflichtend einem Benutzer und einem Projekt zugeordnet.

```
typedef struct _XTHistorydata {
    char szHistory_nr[GUID_LENGTH];       // Eindeutige GUID        [Pflichtfeld]
    char szText[255];                     // Text                   [Pflichtfeld]
    char bText_overflow;
    SYSTEMTIME dtItemdate;                // Eintragsdatum          [Pflichtfeld]
    char szProject_nr[GUID_LENGTH];       // GUID des Projekts      [Pflichtfeld]
    char szUser_nr[GUID_LENGTH];          // GUID des Benutzers     [Pflichtfeld]
    char szCreator_nr[GUID_LENGTH];       // GUID des Erstellers
    SYSTEMTIME dtDatecreated;             // Erstellungsdatum
    SYSTEMTIME dtLastChange;              // Datum/Zeit der letzten Änderung
    char szLocation[255];                 // R7: Ort
    short bLocation_overflow;             // R7: Ist Ort länger als 200 Zeichen?

    char szReserved[256];                 // reserviert für spätere Erweiterungen
} XTHistorydata, *PXTHistorydata, *LPXTHistorydata;
```

## 3.8 Zeitstempel

Ein Zeitstempel enthält einen einzelnen Erfassungsdatensatz, welcher immer einem Benutzer und einem Projekt zugeordnet ist.

```
typedef struct _XTTimestampdata {
     char szTimes_nr[GUID_LENGTH];       // Eindeutige GUID        [Pflichtfeld]
     SYSTEMTIME dtFromTime;              // Datum+Zeit Anfang      [Pflichtfeld]
     SYSTEMTIME dtTillTime;              // Datum+zeit Ende        [Pflichtfeld]
     int iMinutesneeded;                 // Summe der Minuten (wird automatisch
                                            berechnet)
     int iState;                         // Status: 0=xttssAccountNormal,
                                            1=xttssDontAccount,
                                            2=xttssIsAccounted
     short bManualentry;                 // Nachtrag?
     char szComment[255];                // Zeitstempelkommentar
     short bComment_overflow;
     char szProject_nr[GUID_LENGTH];     // GUID des Projekts      [Pflichtfeld]
     char szUser_nr[GUID_LENGTH];        // GUID des Benutzers     [Pflichtfeld]
     char szCreator_nr[GUID_LENGTH];     // GUID des Erstellers
     SYSTEMTIME dtLastChange;            // Datum/Zeit der letzten Änderung
     char szTodo_nr[GUID_LENGTH];        // GUID der Aufgabe
     int iMinutespause;                  // Summe der Pause in Minuten

     char szActivity_nr[GUID_LENGTH];      // R7: GUID der Tätigkeit
     short szPricegroup_nr[GUID_LENGTH];   // R7: GUID der Preisgruppe
     char szLocation[255];                 // R7: Ort
     short bLocation_overflow;             // R7: Ist Ort länger als 200 Zeichen?

     char szReserved[212];               // reserviert für spätere Erweiterungen
} XTTimestampdata, *PXTTimestampdata, *LPXTTimestampdata;
```

## 3.9 Systemdaten

Über die Systemdaten können diverse Konfigurationsparameter ausgelesen werden, die in der weiteren Verarbeitung notwendig bzw. informativ sein könnten.

```
typedef struct _XTSystemdata {
     char szConnectionIniFilename[255]; // Pfad und Name der xperttimer.ini
     char szReportPath[255];            // Pfad auf Reportverz.
     char szDLLFilename[255];           // Pfad auf die verwendete DLL
     char szAppFilename[255];           // Pfad auf das aufrufende Programm
     char szLicenseholder[80];          // Name des Lizenznehmers
     int iPatchlevelDB;                 // Patchlevel der Datenbank
     int iPatchlevelApplication;        // Benötigter Patchlevel der Anwendung
     int iCurrentUserCount;             // Anzahl Benutzer in Datebank
     int iLicenceUserCount;             // Anzahl Benutzer in Lizenzdatei
     int iRevision;                     // Revisinsnr. der DLL
     char szVersion[16];                // Versionsnummer der DLL
     short bModuleActiveHistory;        // Modul „Historie" aktiv?
     short bModuleActiveTodo;           // Modul „Aufgaben" aktiv?
     short bModuleActiveClients;        // Modul „Kunden" aktiv?
     short bModuleActiveDMS;            // Modul „Documents" aktiv?
     short bModuleActiveSync;           // Modul „Synchronisation" aktiv?
     short bModuleActiveReporting;      // Modul „Reporting" aktiv?
     short bModuleActiveInvoice;        // Modul „Rechnungen" aktiv?
     short bModuleActiveProjectItems;   // Modul „Leistungen" aktiv?
     short bModuleActivePricegroups;    // R7: Modul „Preisgruppen" aktiv?
     short bModuleActiveCTI;            // R7: Modul „Telefonie" aktiv?
     char szPricetable_id_nr[GUID_LENGTH]; // R7: GUID der Preisgruppe

     char szReserved[254];              // reserviert für spätere Erweiterungen
} XTSystemdata, *PXTSystemdata, *LPXTSystemdata;
```

# 4 Funktionen

Folgende Funktionen stehen innerhalb der XTAPI.DLL zur Verfügung. Die Notation der Funktionen entspricht hierbei zur Vereinfachung der Pascal-Notation.

## 4.1 Allgemeine Funktionen

### 4.1.1 XTDatabaseConnect

Verbindung mit der Xpert-Timer-Datenbank über die Default-Konfiguration herstellen.

```
function XTDatabaseConnect(bShowConnectionInfo: word;
                           bSilent: word;
                           bAutoLogin: word): word;
```

Parameter:

| | |
|---|---|
| `bShowConnectionInfo` | Zustandsinformationen während dem Herstellen der DB-Verbindung anzeigen. |
| `bSilent` | Keinerlei Fehlermeldungen ausgeben bei Verbindungsfehler. |
| `bAutoLogin` | Automatischer Login des Standardbenutzer. |

Rückgabe:

True wenn die Verbindung herstellt wurde. False wenn es zu Fehlern gekommen ist.

### 4.1.2 XTDatabaseConnectIni

Verbindung mit der Xpert-Timer-Datenbank über die `xperttimer.ini` herstellen.

```
function XTDatabaseConnectIni(szInifilename: PChar;
                              bShowConnectionInfo: word;
                              bSilent: word;
                              bAutoLogin: word): word;
```

Parameter:

| | |
|---|---|
| `szInifilename` | Name der Inidatei welche für die Verbindung verwendet werden soll. |
| `bShowConnectionInfo` | Zustandsinformationen während dem Herstellen der DB-Verbindung anzeigen. |
| `bSilent` | Keinerlei Fehlermeldungen ausgeben bei Verbindungsfehler. |
| `bAutoLogin` | Automatischer Login des Standardbenutzer. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.3 XTDatabaseConnectAccess

Verbindung zu einer MS-Access-Datenbank herstellen.

```
function XTDatabaseConnectAccess(szDBfilename: PChar;
                                 szUsername: Pchar;
                                 szPassword: PChar;
```

```
                                bShowConnectionInfo: word;
                                bSilent: word;
                                bAutoLogin: Boolean): word;
```

Parameter:

| | |
|---|---|
| szDBfilename | Dateiname der MS-Access-Datenbankdatei (.mdb) |
| szUsername | Benutzername für die DB-Anmeldung |
| szPassword | Passwort für die DB-Anmeldung |
| bShowConnectionInfo | Zustandsinformationen während dem Herstellen der DB-Verbindung anzeigen. |
| bSilent | Keinerlei Fehlermeldungen ausgeben bei Verbindungsfehler. |
| bAutoLogin | Automatischer Login des Standardbenutzer. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.4 XTDatabaseConnectSQLite

Verbindung zu einer SQLite-Datenbank herstellen.

```
function XTDatabaseConnectSQLite(szDBfilename: PChar;
                                szUsername: Pchar;
                                szPassword: PChar;
                                bShowConnectionInfo: word;
                                bSilent: word;
                                bAutoLogin: Boolean): word;
```

Parameter:

| | |
|---|---|
| szDBfilename | Dateiname der SQLite-Datenbankdatei (.sqlite) |
| szUsername | Benutzername für die DB-Anmeldung |
| szPassword | Passwort für die DB-Anmeldung |
| bShowConnectionInfo | Zustandsinformationen während dem Herstellen der DB-Verbindung anzeigen. |
| bSilent | Keinerlei Fehlermeldungen ausgeben bei Verbindungsfehler. |
| bAutoLogin | Automatischer Login des Standardbenutzer. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.5 XTDatabaseConnectMSSQL

Verbindung zu einer MS-SQL-Datenbank herstellen.

```
function XTDatabaseConnectMSSQL(szServername,szDBname,szUsername,szPassword: PChar;
                                bSecuritySQL: word;
                                bShowConnectionInfo: word;
                                bSilent: word;
                                bAutoLogin: word): word;
```

Parameter:

| | |
|---|---|
| szServername | Name des MS-SQL-Servers |
| szDBname | Name der Datenbank auf dem SQL-Server |
| szUsername | Benutzername für die DB-Anmeldung |
| szPassword | Passwort für die DB-Anmeldung |
| bSecuritySQL | Anmeldung an SQL-Server über integrierte Windows-Sicherheit vornehmen. |
| bShowConnectionInfo | Zustandsinformationen während dem Herstellen der DB-Verbindung anzeigen. |
| bSilent | Keinerlei Fehlermeldungen ausgeben bei Verbindungsfehler. |
| bAutoLogin | Automatischer Login des Standardbenutzer. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.6 XTDatabaseConnectMySQL

Verbindung zu einer MySQL-Datenbank herstellen.

```
function XTDatabaseConnectMySQL(szServername,szDBname,szUsername,szPassword: PChar;
                                iPort: integer;
                                bUseSSL: word;
                                BShowConnectionInfo: word;
                                BSilent: word;
                                AutoLogin: word): word;
```

Parameter:

| | |
|---|---|
| szServername | Name des MySQL-Servers |
| szDBname | Name der Datenbank auf dem SQL-Server |
| szUsername | Benutzername für die DB-Anmeldung |
| szPassword | Passwort für die DB-Anmeldung |
| iPort | Port des Datenbankservers (z.B. 3306) |
| bUseSSL | SSL-Verschlüsselung für die Verbindung verwenden |
| bShowConnectionInfo | Zustandsinformationen während dem Herstellen der DB-Verbindung anzeigen. |
| bSilent | Keinerlei Fehlermeldungen ausgeben bei Verbindungsfehler. |
| bAutoLogin | Automatischer Login des Standardbenutzer. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.7 XTDatabaseDisconnect

Verbindung mit der Xper-Timer-Datenbank trennen.

```
function XTDatabaseDisconnect():word;
```

Parameter:

Keine

Rückgabe:

True wenn die Verbindung getrennt wurde.

### 4.1.8 XTDatabaseIsConnected

Prüfen ob Verbindung zur DB besteht.

```
function XTDatabaseIsConnected():word;
```

Parameter:

Keine

Rückgabe:

True wenn die Verbindung besteht.

### 4.1.9 XTDatabaseGetIniFilename

Name und Pfad der verwendeten DB-Konfigurationsdatei (XpertTimer.ini) auslesen.

```
function XTDatabaseGetIniFilename(szInifilename: PChar): word;
```

Parameter:

Name und Pfad der Inidatei wird in `inifilename` zurückgegeben. Speicherbedarf 255 Zeichen.

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.10 XTFreeString

Von XTxxxGetList-Funktionen reservierten Stringlisten-Speicher wieder freigeben.

```
procedure XTFreeString(var lpszList: PChar);
```

Parameter:

| | |
|---|---|
| `lpszList` | Zeiger auf freizugebenden Speicherbereich. (Ggf. ist es unter .NET nicht notwendig diesen Speicherbereich freizugeben, da dies vom Speichermanager der Runtime erledigt wird) |

### 4.1.11 XTCreateGUID

Eindeutige Global Unique ID erstellen (GUID). Wird benötigt um mit XT-Primärschlüssel zu arbeiten

```
procedure XTCreateGUID(szGUID: PChar);
```

Parameter:

Erzeugte GUID wird in `szGUID` zurückgegeben. Speicherbedarf 39 Zeichen.

### 4.1.12 XTSendCommand

Fernsteuerungsbefehl an laufende Programminstanz des Xpert-Timers senden.

```
function XTSendCommand (iCommand: integer;szParams: PChar): word;
```

Parameter:

| | |
|---|---|
| iCommand | Befehlscode |
| szParams | Optionaler Parameter z.B. project_nr, user_nr o.ä. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.13 XTSystemGetData

Konfigurationsdaten des XT-Systems abrufen.

```
function XTSystemGetData(systemdata: PSystemRec): word;
```

Parameter:

| | |
|---|---|
| systemdata | Speicherbereich für Rückgabedaten |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.14 XTUserAssignProject

Bestehendes Projekt zu einem Benutzer zuweisen.

```
function XTUserAssignProject(szuser_nr: PChar;
                             szproject_nr: PChar;
                             szparent_project_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szuser_nr | Benutzer dem das Projekt zugewiesen werden soll |
| szproject_nr | Projekt welches zugewiesen werden soll |
| szparent_project_nr | Hauptprojekt welches zugewiesen werden soll |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.15 XTUserUnassignProject

Zugewiesenes Projekt einem Benutzer entziehen.

```
function XTUserUnassignProject(szuser_nr: PChar;
                               szproject_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szuser_nr | Benutzer dem das Projekt entzogen werden soll |
| szproject_nr | Projekt welches entzogen werden soll |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.16 XTUserGetName

Benutzername auslesen

```
function XTUserGetName(szUsername: PChar;
                       szUser_nr: PChar;
                       bIncludePersNr: word): word;
```

Parameter:

| | |
|---|---|
| szUsername | Rückgabewert mit Benutzername. Speicherbedarf 255 Zeichen |
| szUser_nr | Benutzer dessen Name geholt werden soll |
| bIncludePersNr | Soll die Personalnummer ebenfalls ausgegeben werden? |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.17 XTClientGetName

Kundennamen auslesen

```
function XTClientGetName(szClientname: PChar;
                       szClient_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szClientname | Rückgabewert mit Kundenname. Speicherbedarf 255 Zeichen |
| szClient_nr | Kunde dessen Name geholt werden soll |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.1.18 XTProjectGetName

Projektnamen auslesen

```
function XTProjectGetName(szUsername: PChar;
                       szUser_nr: PChar;
                       bIncludeClientname: Boolean): word;
```

Parameter:

| | |
|---|---|
| szProjectname | Rückgabewert mit Projektname. Speicherbedarf 255 Zeichen |
| szProject_nr | Projekt dessen Name geholt werden soll |
| bIncludeClientname | Soll der Name des Kunden ebenfalls ausgegeben werden? |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.2 Benutzeranmeldung

### 4.2.1 XTLogin

Benutzer über Loginname auf Datenbank einloggen.

```
function XTLogin(szLoginname: PChar;
                 bSilent: word): word;
```

Parameter:

| | |
|---|---|
| szLoginname | Loginname des Benutzers |
| bSilent | Keinerlei Fehlermeldungen ausgeben bei Loginfehler. |

Rückgabe:

True wenn die Login ok.

Hinweis:

Generell können sich über die XTAPI nur Benutzer mit Administratorenrechten anmelden. Es gibt jedoch einen Ini-Schalter in der "xperttimer.ini" um von "Nur Admin" auf "Allgemeine Anmeldung" bei der DLL-Benutzung umzuschalten.
[XTAPI]
AdminOnly=0

Desweiteren gibt es die Möglichkeit einzelnen Benutzern das verwenden der XTAPI zu verbieten/erlauben. Dafür gibt es in den Mitarbeitereigenschaften das Feld „XTAPI-Zugriff“

### 4.2.2 XTLoginByUserNr

Benutzer über User_Nr auf Datenbank einloggen.

```
function XTLoginByUserNr(szUser_Nr: PChar;
                 bSilent: word): word;
```

Parameter:

| | |
|---|---|
| szUser_Nr | User_nr des einzuloggenden Benutzers |
| bSilent | Keinerlei Fehlermeldungen ausgeben bei Loginfehler. |

Rückgabe:

True wenn die Login ok.

### 4.2.3 XTLogout

Angemeldeten Benutzer ausloggen.

```
function XTLogout():word;
```

Parameter:

Keine

Rückgabe:

True wenn die Logout ok.

### 4.2.4 XTCurrentUserGetName

Namen des aktuell angemeldeten Benutzers abfragen.

```
procedure XTCurrentUserGetName (szUsername: PChar);
```

Parameter:

Name und Vorname wird in `szUsername` zurückgegeben. Speicherbedarf 81 Zeichen.

### 4.2.5 XTCurrentUserGetUserNr

GUID (User_Nr) des aktuell angemeldeten Benutzers abfragen.

```
procedure XTCurrentUserGetUserNr (szUser_Nr: PChar);
```

Parameter:

User_Nr wird in `szUser_Nr` zurückgegeben. Speicherbedarf 39 Zeichen.

### 4.2.6 XTCurrentUserGetRunningProjectNr

GUID (Project_Nr) des laufenden Projekts des angemeldeten Benutzers abfragen.

```
procedure XTCurrentUserGetRunningProjectNr (szProject_Nr: PChar);
```

Parameter:

Project_Nr wird in `szProject_Nr` zurückgegeben. Speicherbedarf 39 Zeichen.

## 4.3 Datenobjekte abfragen

Über die folgenden Funktionen können Datensätze des Xpert-Timers in den Speicher geladen werden.

### 4.3.1 XTUserGetData

Benutzerdatensatz einlesen.

```
function XTUserGetData(szuser_nr: PChar; userdata: PUserRec): word;
```

Parameter:

| | |
|---|---|
| szuser _nr | GUID des Benutzers |
| userdata | Speicherbereich für Rückgabedaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.3.2 XTClientGetData

Kundendatensatz einlesen.

```
function XTClientGetData(szclient_nr: PChar; clientdata: PClientRec): word;
```

Parameter:

| | |
|---|---|
| szClient_nr | GUID des Kunden |
| clientdata | Speicherbereich für Rückgabedaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.3.3 XTProjectGetData

Projektdatensatz einlesen.

```
function XTProjectGetData(szproject_nr: PChar; projectdata: PProjectRec): word;
```

Parameter:

| | |
|---|---|
| szproject_nr | GUID des Projekts |
| projectdata | Speicherbereich für Rückgabedaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.3.4 XTHistoryGetData

Historieneintrag einlesen.

```
function XTHistoryGetData(szhistory_nr: PChar; historydata: PHistoryRec): word;
```

Parameter:

| | |
|---|---|
| szhistory_nr | GUID des Eintrags |
| historydata | Speicherbereich für Rückgabedaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.3.5 XTTaskGetData

Aufgabendatensatz einlesen.

```
function XTTaskGetData(sztodo_nr: PChar; tododata: PTodoRec): word;
```

Parameter:

| | |
|---|---|
| sztodo_nr | GUID der Aufgabe |
| tododata | Speicherbereich für Rückgabedaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.3.6 XTTimestampGetData

Zeitstempeldatensatz einlesen.

```
function XTTimestampGetData(sztimes_nr: PChar;
                            timestampdata: PTimestampRec): word;
```

Parameter:

| | |
|---|---|
| sztimes_nr | GUID des Zeitstempels |
| timestampdata | Speicherbereich für Rückgabedaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.4 Datenobjekte schreiben/ändern

Über die folgenden Funktionen können geänderte Datensätze des Xpert-Timers zurück in die Datenbank geschrieben werden.

### 4.4.1 XTUserSetData

Benutzerdatensatz schreiben.

```
function XTUserSetData(userdata: PUserRec): word;
```

Parameter:

| | |
|---|---|
| `userdata` | Speicherbereich mit Benutzerdaten |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.4.2 XTClientSetData

Kundendatensatz schreiben.

```
function XTClientSetData(clientdata: PClientRec): word;
```

Parameter:

| | |
|---|---|
| `clientdata` | Speicherbereich mit Kundendaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.4.3 XTProjectSetData

Projektdatensatz schreiben.

```
function XTProjectSetData(projectdata: PProjectRec): word;
```

Parameter:

| | |
|---|---|
| `projectdata` | Speicherbereich mit Projektdaten |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.4.4 XTHistorySetData

Historieneintrag schreiben.

```
function XTHistorySetData(historydata: PHistoryRec): word;
```

Parameter:

| | |
|---|---|
| historydata | Speicherbereich mit Eintragsdaten. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.4.5 XTTaskSetData

Aufgabendatensatz schreiben.

```
function XTTaskSetData(tododata: PTodoRec): word;
```

Parameter:

| | |
|---|---|
| tododata | Speicherbereich mit Aufgabendaten |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.4.6 XTTimestampSetData

Zeitstempeldatensatz schreiben.

```
function XTTimestampSetData(timestampdata: PTimestampRec): word;
```

Parameter:

| | |
|---|---|
| timestampdata | Speicherbereich mit Zeitstempeldaten |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.5 Datensatzlisten einlesen

Über die folgenden Funktionen können Listen von Datenobjekten eingelesen werden. Auf diese Weise können die in der Datenbank enthaltenen Datensätze aufgelistet und ggf. anhand ihrer eindeutigen Nummer (GUID) weiterverarbeitet werden.

Der für die Rückgabemenge erforderliche Speicherbereich wird hierbei dynamisch reserviert und muss nach der Verarbeitung der Datenliste wieder mit der Prozedur „XTFreeMem“ freigegeben werden.

### 4.5.1 XTUserGetList

Liste aller Benutzer abrufen.

```
function XTUserGetList(var szUserlist: PChar):integer;
```

Parameter:

| | |
|---|---|
| szUserlist | Speicherbereich auf Rückgabeliste |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
user_nr;firstname;lastname;loginname;persnr;active
```

Beispiel:

```
01. {0767F4E9-CC24-4D6A-87DA-F6D9B088D503};Maier;Max;Maxi;1;1
02. {1BE5F43B-96DB-402D-9288-C0D7ACDE9940};Demo;Dieter;Diez;2;1
```

### 4.5.2 XTClientGetList

Liste aller Kunden abrufen.

```
function XTClientGetList(var szClientlist: PChar):integer;
```

Parameter:

| | |
|---|---|
| szClientlist | Speicherbereich auf Rückgabeliste |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
client_nr;clientname;clientid;active
```

Beispiel:

```
01. {923C02D4-EA40-4F28-8630-58D729AB6EDD};Schreinerei Schobel;1;1
02. {F997C403-C76B-495D-903F-3810846FC683};Autohaus Maier;2;1
```

### 4.5.3 XTProjectGetList

Projektliste abrufen.

```
function XTProjectGetList(var szProjectlist: PChar;
                          szuser_nr: PChar;
                          szclient_nr: PChar;
                          szparent_project_nr: PChar):integer;
```

Parameter:

| | |
|---|---|
| szProjectlist | Speicherbereich auf Rückgabeliste |
| szuser_nr | Filter auf bestimmten Benutzer (NULL wenn alle Benutzer) |
| szclient_nr | Filter auf bestimmten Kunden (NULL wenn alle Kunden) |
| szparent_project_nr | Filter auf bestimmtes Hauptprojekt (NULL wenn alle Projekte) |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
project_nr;projectname;parentprojectname;clientname;parent_project_nr;client_nr;
state;projectnumber;minutesneeded;minutesestimated
```

Beispiel:

```
01. {923C02D4-EA40-4F28-8630-58D729AB6EDD};Unterprojekt1;Hauptprojekt1;Kunde1;
    {F997C403-C76B-495D-903F-3810846FC683};{923C02D4-EA40-4F28-8630-
    58D729AB6EDD};0;4711;90;120
```

### 4.5.4 XTTaskGetList

Liste der Aufgaben abrufen.

```
function XTTaskGetList(var szTasklist: PChar;
                       szuser_nr: PChar;
                       szclient_nr: PChar;
                       szproject_nr: PChar;
                       itaskstate: integer;
                       dtduefrom:Systemtime;
                       dtduetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szClientlist | Speicherbereich auf Rückgabeliste |
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| itaskstate | Filter auf Aufgabenstatus |

| | |
|---|---|
| dtduefrom | Filter auf Fälligkeitsdatum |
| dtduetill | Filter auf Fälligkeitsdatum |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
todo_nr;user_nr;project_nr;taskstate;todonumber;date_created;date_start;date_due;
date_done;date_read;subject;minutesneeded;minutesestimated
```

Beispiel:

```
01.{923C02D4-EA40-4F28-8630-58D729AB6EDD};{F997C403-C76B-495D-903F-
   3810846FC683};{923C02D4-EA40-4F28-8630-58D729AB6EDD};0;7;2010-04-20;2010-04-24;
   2010-04-26;1899-12-30;2010-04-25;Aufgabe 1;60;50
```

Ein nicht gesetztes Datum, wie hier im Beispiel der Wert „date_done“ wird mit dem Datumswert 30.12.1899 belegt.

### 4.5.5 XTHistoryGetList

Liste der Historieneinträge abrufen.

```
function XTHistoryGetList(var szHistorylist: PChar;
                          szuser_nr: PChar;
                          szclient_nr: PChar;
                          szproject_nr: PChar;
                          dtdatefrom:Systemtime;
                          dtdatetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szHistorylist | Speicherbereich auf Rückgabeliste |
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| dtdatefrom | Filter auf Eintragsdatum |
| dtdatetill | Filter auf Eintragsdatum |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
history_nr;user_nr;project_nr;date;text
```

Beispiel:

```
01.{923C02D4-EA40-4F28-8630-58D729AB6EDD};{F997C403-C76B-495D-903F-
   3810846FC683};{923C02D4-EA40-4F28-8630-58D729AB6EDD};2010-04-20;Eintrag1
```

### 4.5.6 XTTimestampGetList

Liste der Zeitstempel abrufen.

```
function XTTimestampGetList(var szTimestamplist: PChar;
                            szuser_nr: PChar;
                            szclient_nr: PChar;
                            szproject_nr: PChar;
                            sztodo_nr: PChar;
                            istate: integer;
                            dtdatefrom:Systemtime;
                            dtdatetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szTimestamplist | Speicherbereich auf Rückgabeliste |
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| sztodo_nr | Filter auf Aufgabe |
| istate | Filter auf Zeitstempelstatus |
| dtdatefrom | Filter auf Zeitstempeldatum |
| dtdatetill | Filter auf Zeitstempeldatum |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
times_nr;user_nr;project_nr;todo_nr;FromTime;TillTime;MinutesNeeded;MinutesPause;
state;ManualEntry;Comment
```

Beispiel:

```
01. {923C02D4-EA40-4F28-8630-58D729AB6EDD};{F997C403-C76B-495D-903F-
    3810846FC683};{923C02D4-EA40-4F28-8630-58D729AB6EDD};{789A02B4-AE40-F482-8630-
    58D729AB6DEE};2010-04-20 10:40;2010-04-20 11:20;40;5;0;N;Projektzeichnung
    erstellt
```

### 4.5.7 XTProjecttypeGetList

Liste der Projekttypen abrufen.

```
function XTProjecttypeGetList(var szProjecttypelist: PChar):integer;
```

Parameter:

| | |
|---|---|
| szProjecttypelist | Speicherbereich auf Rückgabeliste |

Rückgabe:

Anzahl der Einträge der Datenliste.

Die Datensätze werden zeilenweise in folgendem Format zurückgeliefert:

```
projecttype_nr;typename
```

Beispiel:

```
01.{923C02D4-EA40-4F28-8630-58D729AB6EDD};Entwicklung
```

## 4.6 Datensätze anlegen

Über die folgenden Funktionen können neue Datensätze erstellt werden.

### 4.6.1 XTUserAdd

Neuen Benutzer anlegen.

```
function XTUserAdd(szuser_nr: PChar;
                   szloginname: PChar;
                   szfirstname: PChar;
                   szlastname: PChar): word;
```

Parameter:

| | |
|---|---|
| szuser_nr | Rückgabewert der GUID des neuen Benutzers |
| szloginname | Anmeldename des neuen Benutzers für Login |
| szfirstname | Vorname des Benutzers |
| szlastname | Nachname des Benutzers |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.6.2 XTProjectAdd

Neues Projekt anlegen.

```
function XTProjectAdd(szproject_nr: PChar;
                      szprojectname: PChar;
                      szclient_nr: PChar;
                      szparent_project_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szproject_nr | Rückgabewert der GUID des neuen Projekts |
| szprojectname | Name des neuen Projekts |
| szclient_nr | GUID des zugeordneten Kunden ("{11111111-1111-1111-1111-111111111111}" wenn Projekt keinem Kunden zugeordnet werden soll) |
| szparent_project_nr | GUID des Hauptprojekts |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.6.3 XTClientAdd

Neuen Kunden anlegen.

```
function XTClientAdd(szclient_nr: PChar;
                      szclientname: PChar): word;
```

Parameter:

| | |
|---|---|
| szclient_nr | Rückgabewert der GUID des neuen Kunden |
| szclientname | Name des neuen Kunden |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.6.4 XTTaskAdd

Neue Aufgabe anlegen.

```
function XTTaskAdd(sztodo_nr: PChar;
                   szuser_nr: PChar;
                   szproject_nr: PChar;
                   szsubject: PChar;
                   sztext: PChar): word;
```

Parameter:

| | |
|---|---|
| sztodo_nr | Rückgabewert der GUID des neuen Projekts |
| szuser_nr | Benutzer für den die Aufgabe erstellt werden soll |
| szproject_nr | Projekt an das die Aufgabe angehängt werden soll |
| szsubject | Betreff der Aufgabe |
| sztext | Text der Aufgabe (max. 255 Zeichen) |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.6.5 XTHistoryAdd

Neuen Historieneintrag anlegen.

```
function XTHistoryAdd(szhistory_nr: PChar;
                      szuser_nr: PChar;
                      szproject_nr: PChar;
                      sztext: PChar): word;
```

Parameter:

| | |
|---|---|
| szhistory_nr | Rückgabewert der GUID des neuen Eintrags |
| szuser_nr | Benutzer für den der Eintrag erstellt werden soll |
| szproject_nr | Projekt an das der Eintrag angehängt werden soll |
| sztext | Text des Eintrags (max. 255 Zeichen) |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.6.6 XTTimestampAdd

Neue Zeitstempel anlegen.

```
function XTTimestampAdd(sztimes_nr: PChar;
                        szuser_nr: PChar;
                        szproject_nr: PChar;
                        sztodo_nr: PChar;
                        dtFromTime:Systemtime;
                        dtTillTime: Systemtime;
                        szcomment: PChar): word;
```

Parameter:

| | |
|---|---|
| sztimes_nr | Rückgabewert der GUID des neuen Zeitstempels |
| szuser_nr | Benutzer für den der Eintrag erstellt werden soll |
| szproject_nr | Projekt für das der Zeitstempel angehängt werden soll |
| sztodo_nr | Aufgabe an die der Zeitstempel angehängt werden soll |
| dtFromTime | Zeitstempelanfang |
| dtTillTime | Zeitstempelende |
| szcomment | Kommentar zum Zeitstempel (max. 255 Zeichen) |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.7 Datensätze löschen

Über die folgenden Funktionen können bestehende Datensätze in der Datenbank gelöscht werden.

Hinweis: Datensätze werden innerhalb der XT-Datenbank niemals wirklich gelöscht, sondern nur als „gelöscht“ gekennzeichnet.

### 4.7.1 XTUserDelete

Benutzer löschen.

```
function XTUserDelete(szuser_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szuser_nr | GUID des zu löschenden Benutzers |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.7.2 XTClientDelete

Kunde löschen.

```
function XTClientDelete(szclient_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szclient_nr | GUID des zu löschenden Kunden |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.7.3 XTProjectDelete

Projekt löschen.

```
function XTProjectDelete(szproject_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szproject_nr | GUID des zu löschenden Projekts |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.7.4 XTTaskDelete

Aufgabe löschen.

```
function XTTaskDelete(sztodo_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| sztodo_nr | GUID der zu löschenden Aufgabe |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.7.5 XTHistoryDelete

Historieneintrag löschen.

```
function XTHistoryDelete(szhistory_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| szhistory_nr | GUID des zu löschenden Eintrags |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.7.6 XTTimestampDelete

Zeitstempel löschen.

```
function XTTimestampDelete(sztimes_nr: PChar): word;
```

Parameter:

| | |
|---|---|
| sztimes_nr | GUID des zu löschenden Zeitstempels |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.8 Auswertungsfunktionen

Über die folgenden Funktionen können verschiedene Auswertungen über die erfassten Daten erzeugt werden.

### 4.8.1 XTCalcTotalMinutes

Benötigte Zeit eines Projekts abrufen.

```
function XTCalcTotalMinutes(szuser_nr: PChar;
                            szclient_nr: PChar;
                            szproject_nr: PChar;
                            sztodo_nr: PChar;
                            istate: integer;
                            dtdatefrom: Systemtime;
                            dtdatetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| sztodo_nr | Filter auf Aufgabe |
| istate | Filter auf Zeitstempelstatus |
| szdatefrom | Filter auf Zeitstempeldatum |
| szdatetill | Filter auf Zeitstempeldatum |

Rückgabe:

Anzahl der benötigten Minuten.

### 4.8.2 XTRecalcProjectTotals

Benötigte Zeit eines Projekts aus den einzelnen Zeitstempeln nachberechnen.

```
function XTRecalcProjectTotals(szproject_nr: PChar;
                               sztodo_nr: PChar):integer;
```

Parameter:

| | |
|---|---|
| szproject_nr | Projekt das nachberechnet werden soll |
| sztodo_nr | Aufgabe die nachberechnet werden soll |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.8.3 XTRecalcProjectTotalsAll

Benötigte Zeit aller Projekte (eines Kunden) aus den einzelnen Zeitstempeln nachberechnen.

```
function XTRecalcProjectTotalsAll(szclient_nr: PChar):integer;
```

Parameter:

| | |
|---|---|
| szclient_nr | Kunde der nachberechnet werden soll. Wenn leer, dann alle Kunden und Projekte. |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.8.4 XTTimestampPrintList

Druckansicht für Zeitstempelreport aufrufen.

```
function XTTimestampPrintList(szuser_nr: PChar;
                              szclient_nr: PChar;
                              szproject_nr: PChar;
                              sztodo_nr: PChar;
                              istate: integer;
                              dtdatefrom: Systemtime;
                              dtdatetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| sztodo_nr | Filter auf Aufgabe |
| istate | Filter auf Zeitstempelstatus |
| dtdatefrom | Filter auf Zeitstempeldatum |
| dtdatetill | Filter auf Zeitstempeldatum |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.8.5 XTTimestampExportList

Zeitstempelliste exportieren in CSV, XML, Excel oder als Text in die Zwischenablage.

```
function XTTimestampExportList(szFilename: PChar;
                               iexportmode: integer;
                               bexportguids: integer;
                               szuser_nr: PChar;
                               szclient_nr: PChar;
                               szproject_nr: PChar;
                               sztodo_nr: PChar;
                               istate: integer;
                               dtdatefrom: Systemtime;
                               dtdatetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szFilename | Dateiname der Ausgabedatei |
| iexportmode | Exportmodus: 0 = CSV, 1 = XML, 2 = Excel, 3 = Clipboard |
| bexportguids | Sollen die GUIDs der Datensätze ebenfalls exportiert werden? |
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| sztodo_nr | Filter auf Aufgabe |
| istate | Filter auf Zeitstempelstatus |
| dtdatefrom | Filter auf Zeitstempeldatum |
| dtdatetill | Filter auf Zeitstempeldatum |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.8.6 XTPrintReport

Beliebige Reportdatei drucken.

```
function XTPrintreport(szFilename: PChar;
                       bpreview: integer;
                       szuser_nr: PChar;
                       szclient_nr: PChar;
                       szproject_nr: PChar;
                       dtdatefrom:Systemtime;
                       dtdatetill: Systemtime):integer;
```

Parameter:

| | |
|---|---|
| szFilename | Dateiname der Reportdatei |
| bpreview | Sollen die GUIDs der Datensätze ebenfalls exportiert werden? |
| szuser_nr | Filter auf Benutzer |
| szclient_nr | Filter auf Kunden |
| szproject_nr | Filter auf Projekt |
| dtdatefrom | Filter auf Zeitstempeldatum |
| dtdatetill | Filter auf Zeitstempeldatum |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.9 Auswahldialoge

Über die folgenden Funktionen können Stammdatenobjekte über einen komfortablen Auswahldialog mit Suchfunktion ausgewählt werden.

### 4.9.1 XTSelectUser

Benutzerauswahl anzeigen.

```
function XTSelectUser(szuser_nr: PChar;
                      szsearch: PChar;
                      sztitle: PChar;
                      sztitletext: PChar): word;
```

Parameter:

| | |
|---|---|
| szuser_nr | Rückgabewert: GUID des gewählten Benutzers. Durch setzen der user_nr vor dem Öffnen des Auswahldialogs kann bereits ein Benutzer in der Benutzerliste vorselektiert werden. |
| szsearch | Suchbegriff als Vorgabewert im Auswahldialog |
| sztitle | Fenstertitel des Dialogs. |
| sztitletext | Beschreibungstext zu Auswahlvorgang. Z.B. „Wählen Sie hier den Benutzer für den Sie die Auswertung vornehmen möchten.“ |

Rückgabe:

True wenn Benutzer über OK ausgewählt. False bei Abbruch.

### 4.9.2 XTSelectClient

Kundenauswahl anzeigen.

```
function XTSelectClient(szclient_nr: PChar;
                        szsearch: PChar;
                        sztitle: PChar;
                        sztitletext: PChar): word;
```

Parameter:

| | |
|---|---|
| szclient_nr | Rückgabewert: GUID des gewählten Kunden. Durch setzen der client_nr vor dem Öffnen des Auswahldialogs kann bereits ein Kunde in der Kundenrliste vorselektiert werden. |
| szsearch | Suchbegriff als Vorgabewert im Auswahldialog |
| sztitle | Fenstertitel des Dialogs. |
| sztitletext | Beschreibungstext zu Auswahlvorgang. Z.B. „Wählen Sie hier den Kunden für den Sie die Auswertung vornehmen möchten.“ |

Rückgabe:

True wenn Kunde über OK ausgewählt. False bei Abbruch.

### 4.9.3 XTSelectProject

Projektauswahl anzeigen.

```
function XTSelectProject(szproject_nr: PChar;
                         szuser_nr: PChar;
                         szclient_nr: PChar;
                         szsearch: PChar;
                         sztitle: PChar;
                         sztitletext: PChar): word;
```

Parameter:

| | |
|---|---|
| szproject_nr | Rückgabewert: GUID des gewählten Projekts. Durch setzen der project_nr vor dem Öffnen des Auswahldialogs kann bereits eine Vorauswahl im Dialog getroffen werden. |
| szuser_nr | Optionaler Rückgabewert: GUID des in den Filtereinstellungen gewählten Benutzers. Durch setzen der user_nr vor dem Öffnen des Auswahldialogs kann bereits eine Vorauswahl im Dialog getroffen werden. |
| szclient_nr | Optionaler Rückgabewert: GUID des gewählten Kunden. Durch setzen der client_nr vor dem Öffnen des Auswahldialogs kann bereits eine Vorauswahl im Dialog getroffen werden. |
| szsearch | Suchbegriff als Vorgabewert im Auswahldialog |
| sztitle | Fenstertitel des Dialogs. |
| sztitletext | Beschreibungstext zu Auswahlvorgang. Z.B. „Wählen Sie hier das Projekt für das Sie die Auswertung vornehmen möchten.“ |

Rückgabe:

True wenn Benutzer über OK ausgewählt. False bei Abbruch.

## 4.10 Suchfunktionen

### 4.10.1 XTFindUserGUIDByNr

Benutzer anhand seiner Personalnummer finden.

```
function XTFindUserGUIDByNr(szUser_nr: PChar;
                            ipersnr: integer): word;
```

Parameter:

| | |
|---|---|
| szUser_Nr | Rückgabewert mit user_nr. Speicherbedarf 39 Zeichen |
| ipersnr | Personalnummer des gesuchten Benutzers |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.10.2 XTFindClientGUIDByNr

Kunde anhand seiner Kundennummer finden.

```
function XTFindClientGUIDByNr(szClient_nr: PChar;
                              iclientid: integer): word;
```

Parameter:

| | |
|---|---|
| szClient_Nr | Rückgabewert mit client_nr. Speicherbedarf 39 Zeichen |
| iclientid | Kundennummer des gesuchten Kunden |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.10.3 XTFindProjectGUIDByNr

Projekt anhand seiner Projektnummer finden.

```
function XTFindProjectGUIDByNr(szProject_nr: PChar;
                               szProjectnumber: PChar): word;
```

Parameter:

| | |
|---|---|
| szProject_Nr | Rückgabewert mit project_nr. Speicherbedarf 39 Zeichen |
| szProjectnumber | Alphanummerische Projektnummer des gesuchten Projekts |

Rückgabe:

True/False bei Erfolg/Fehler.

### 4.10.4 XTFindTaskGUIDByNr

Aufgabe anhand seiner Aufgabennummer finden.

```
function XTFindTaskGUIDByNr(  szTodo_nr: PChar;
                              iTodoId: integer): word;
```

Parameter:

| | |
|---|---|
| szTodo_Nr | Rückgabewert mit todo_nr. Speicherbedarf 39 Zeichen |
| iTodoId | Aufgabennummer der gesuchten Aufgabe |

Rückgabe:

True/False bei Erfolg/Fehler.

## 4.11 Hilfsfunktionen

### 4.11.1 XTStringToSystemtime

Einen String im Format „YYYY-MM-DD HH:MM“ in eine Systemtime wandeln.

```
procedure XTStringToSystemtime(szDatetime: PChar;
                               dtDatetime: Systemtime);
```

Parameter:

| | |
|---|---|
| szDatetime | Datum/Zeit |
| dtDatetime | Rückgabewert |

### 4.11.2 XTSystemtimeToDateTimeString

Eine Systemtime in einen String im Format „YYYY-MM-DD HH:MM“ wandeln.

```
procedure XTSystemtimeToDateTimeString(dtDatetime: Systemtime;
                                       szDatetime: PChar);
```

Parameter:

| | |
|---|---|
| dtDatetime | Datum/Zeit |
| szDatetime | Rückgabewert |

### 4.11.3 XTSystemtimeToDateString

Eine Systemtime in einen String im Format „YYYY-MM-DD“ wandeln.

```
procedure XTSystemtimeToDateString(dtDatetime: Systemtime;
                                   szDate: PChar);
```

Parameter:

| | |
|---|---|
| dtDatetime | Datum |
| szDatetime | Rückgabewert |

# 5 Beispiele

Folgende, in Visual Studio 2010 unter C++ erstellte, Codebeispiele sollen das Zusammenspiel der einzelnen XT-Funktionen erläutern. Die Beispiele sind als Konsolenanwendungen gestaltet um unnötigen Overhead zu vermeiden.

Weitere Beispiele zu anderen Programmierumgebungen finden Sie im Ordner „\Examples\“.

```
#include "stdafx.h"
#include "XTAPI.h"
#include <windows.h>
#include <stdio.h>
#include <iostream>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    LPSTR lpstrInifile;
    LPSTR lpstrUsername;
    LPSTR lpstrActiveproject;
    LPSTR lpstrUserlist;
    SYSTEMTIME stDateFrom;
    SYSTEMTIME stDateTill;
    LPSTR lpstrDateFrom;
    LPSTR lpstrDateTill;


    XTProjectdata projectdata;

    // XTAPI.DLL laden
    if (XTLoadDLL())
    {
       XTInitialize();
       // Pfad+Name der Konfigurationsdatei "xperttimer.ini" holen
        lpstrInifile = (LPSTR)malloc(255);
       cout << "XTDatabaseGetIniFilename " << endl;
       XTDatabaseGetIniFilename(lpstrInifile);
       cout << " " << lpstrInifile << endl;
        free(lpstrInifile);

          // Datenbankverbindung aufbauen
       cout << "XTDatabaseConnect " << endl;
       if (XTDatabaseConnect(XTC_TRUE, XTC_FALSE, XTC_TRUE) <> XTC_FALSE) {
         // Name des auf diesem System angemeldeten Benutzers holen
         lpstrUsername = (LPSTR)malloc(255);
         cout << "XTCurrentUserGetName " << endl;
         XTCurrentUserGetName(lpstrUsername);
         cout << "CurrentUsername: " << lpstrUsername << endl;
         free(lpstrUsername);

         cout << "<< Press key to continue >>" << endl; _getch();

         // Liste aller Benutzer abfragen
         XTUserGetList(lpstrUserlist);
         cout << "XTUserGetList: " << endl;
         cout << lpstrUserlist << endl;
          XTFreeMem(lpstrUserlist);

         // Liste aller Zeitstempel für 2011 abfragen
         lpstrTimestamplist = NULL;
         XTStringToSystemtime("2011-01-01",&stDateFrom);
         XTStringToSystemtime("2011-12-31",&stDateTill);
         lpstrDateFrom = (LPSTR)malloc(20);
         lpstrDateTill = (LPSTR)malloc(20);
         XTSystemtimeToDateString(&stDateFrom, lpstrDateFrom);
         XTSystemtimeToDateString(&stDateTill, lpstrDateTill);

         XTTimestampGetList(lpstrTimestamplist,NULL,NULL,NULL,NULL,0,stDateFrom,stDateTill);
         cout << "XTTimestampGetList: " << lpstrDateFrom << " - " << lpstrDateTill << endl;
         cout << lpstrTimestamplist << endl;
        XTFreeString(lpstrTimestamplist);
```

```
          free(lpstrDateFrom);
          free(lpstrDateTill);

          // Aktuell laufendes Projekt abfragen
          lpstrActiveproject = (LPSTR)malloc(39);
          XTCurrentUserGetRunningProjectNr(lpstrActiveproject);
          cout << "Activeproject_Nr: " << lpstrActiveproject << endl;
          if (strlen(lpstrActiveproject) > 0)
          {
            cout << "XTProjectGetData " << endl;
            XTProjectGetData(lpstrActiveproject,&projectdata);
            cout << "Activeproject: " << projectdata.projectname << endl;

            cout << "<< Press key continue >>" << endl; _getch();

             // Laufendes Projekt stoppen
            cout << "XTSendCommand: Stop " << endl;
            XTSendCommand(XTCMD_STOP,NULL);

            cout << "<< Press key continue >>" << endl; _getch();

             // Letztes Projekt wieder starten
            cout << "XTSendCommand: Start " << lpstrActiveproject << endl;
            XTSendCommand(XTCMD_START,lpstrActiveproject);
          }

           free(lpstrActiveproject);
           // Datenbankverbindung trennen
          cout << "XTDatabaseDisconnect " << endl;
           XTDatabaseDisconnect;
    }
    else
      cout << " FAILED! " << endl;
  }

  // DLL freigeben
  XTUnloadDLL();

  cout << "<< Press key to exit >>" << endl; _getch();
  return 0;
}
```

# 6 Dokumenthistorie

Hier werden alle an diesem Dokument vorgenommenen Änderungen chronologisch festgehalten.

| Datum | Beschreibung |
|---|---|
| 01.06.2010 | Erste Version der Dokumentation |
| 28.06.2011 | Beispiele Erweitert |
| 04.08.2011 | **Revision 3**<br>• Beispiele Erweitert<br>• Umstellung der DLL für Zugriff auf XT2.7 (Unicode)<br>• Bugfix: Fehler im Speicherlayout behoben<br>• Bugfix: TimestampAdd gibt nun keine Fehlermeldungen mehr aus und liefert XTC_False w<br>Zeitstempel bereits erfasst wurde. |
| 23.11.2011 | • Bugfix: Funktion "TimestampSetData" hat Feld "itemdate" nicht mit dem Tag des Zeitstemp<br>belegt. |
| 24.07.2012 | **Revision 4**<br>• Beispiele Erweitert<br>• Umstellung der DLL für Zugriff auf XT3.0<br>• Struktur " XTTaskdata" erweitert um " szTaskstatecomment" und "bTaskstatecomment_ove<br>• Struktur " XTTimestampdata" erweitert um " szTodo_nr" und "iMinutespause"<br>• XTTaskGetList: Ausgabeformat erweitert um "MinutesNeeded" und "MinutesEstimated"<br>• XTTimestampGetList: Parameterliste erweitert um "todo_nr"<br>• XTTimestampGetList: Ausgabeformat erweitert um "todo_nr" und "minutespause"<br>• XTUserGetList: Ausgabeformat erweitert um "active"<br>• XTClientGetList: Ausgabeformat erweitert um "active"<br>• XTProjectGetList: Ausgabeformat erweitert um "minutesestimated"<br>• XTUserAssignProject: Standardmäßig wird nun das Projekthäkchen für den betreffenden U<br>automatisch gesetzt um das Projekt im Menü anzuzeigen.<br>• XTCalcTotalMinutes: Parameterliste erweitert um "todo_nr"<br>• XTTimestampPrintList: Parameterliste erweitert um "todo_nr"<br>• XTTimestampExportList: Parameterliste erweitert um "todo_nr"<br>• XTTimestampAdd: Parameterliste erweitert um "todo_nr"<br>• Neue Funktion XTProjecttypeGetList um die Liste der vorhandenen Projekttypen abfragen<br>können<br>• Neue Funktion " XTFindTaskGUIDByNr" um Aufgabe anhand der Aufgabennummer finden<br>können<br>Neue Funktion " XTRecalcProjectTotals" um Projektsummen aus den einzelnen Zeitstempe<br>nachberechnen zu können<br>• Neue Funktion " XTRecalcProjectTotalsAll" um alle Projektsummen aus den einzelnen<br>Zeitstempeln nachberechnen zu können |

| | |
|---|---|
| 21.08.2012 | **V3.0.1.847**<br>• Beispiele Erweitert<br><br>• Bugfix: Fehler beim Anlegen einer Kontaktadresse zu einem Kunden<br>• Clientcontact: Neues Feld "firstname"<br>• Clientcontact: Feld "title" ist jetzt vom Typ string[20] |
| 23.08.2012 | **V3.0.1.848**<br>• Beispiele Erweitert<br><br>• Projectdata: Neue Felder<br>isinterrupt, blinking, noidlepause , addhistorymode<br>addtimestampmode, addhistorytext , addtimestamptext , subprojectorder<br>submenu , archived |
| 27.03.2013 | **V3.0.3.857**<br>• Bugfix: Anlegen einer Aufgabe mit TaskAdd() trägt Subject nicht korrekt ein<br>• MySQLConnect: Erweiterung der Funktionsparameter um Port und SSL<br>function XTDatabaseConnectMySQL(<br>szServername,szDBname,szUsername,szPassword: PAnsiChar;<br>iPort: Integer; bUseSSL: Word;<br>bShowConnectionInfo: Word;bSilent: Word;bAutoLogin: Word<br>):Word; |
| 11.01.2014 | **Revision 5: V4.0.0.941**<br>• Beispiele Erweitert<br><br>• Umstellung der DLL für Zugriff auf XT4.0<br><br>• Bugfix: Fehler beim Export von Zeitstempeln in CSV-Format. |
| 01.04.2015 | **Revision 5: V4.5.8.981**<br>• Umstellung der DLL für Zugriff auf XT4.5 |
| 25.02.2016 | **Revision 6: V5.0.2.1008**<br>• Umstellung der DLL für Zugriff auf XT5.0 |
| 31.08.2017 | **Revision 6: V5.0.9.1026**<br>• Bugfix: Bei auslesen eines Zeitstempels mit XTAPITimestampGetData wurde das Feld „lastchange“ nicht berücksichtigt |
| 08.09.2017 | **Revision 7: V6.0.0.1050**<br>• Umstellung der DLL für Zugriff auf XT6.0<br>• Erweiterung der Strukturen um Felder der V6<br><br>• XTUserdata; `szMainworkgroup_nr, szPricegroup_nr`<br><br>• XTSystemdata: `bModuleActivePricegroups, bModuleActiveCTI, szPricetable_id_nr`<br><br>• XTTimestampdata: `szActivity_nr, szPricegroup_nr, szLocation, bLocation_overfl`<br><br>• XTHistorydata: `szLocation, bLocation_overflow`<br><br>• XTTaskdata: `szUserdefined1` |


- szUserdefined5

- XTProjectData: szSecondincharge_nr, szMaincontact_nr, szCreated_workgroup_nr, szPricetable_id_nr, szUserdefined1 - szUserdefined10

- XTClientdata: szUserdefined1 - szUserdefined10