

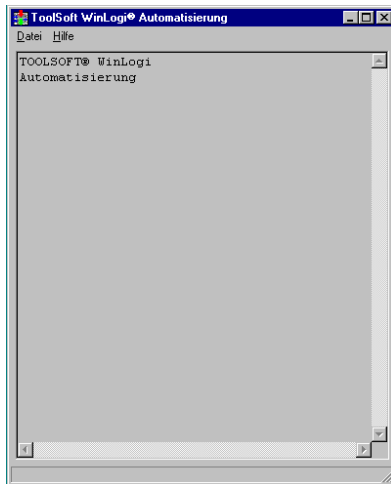
# Bedienung LogiAuto

## Einleitung

Das Modul LogiAuto der Applikation WinLogi wird als Schnittstelle für die OLE-Automation mit anderen Applikationen benutzt. Diese Applikationen erhalten damit Zugriff auf die Daten der Applikation WinLogi. Dabei können Daten abgefragt und Daten gespeichert werden. Nachfolgend erhalten Sie eine kurze Einführung in das Modul LogiAuto.

## Allgemeine Beschreibung

Das Modul LogiAuto ist als OLE-Automation Server konzipiert. Sie können das Modul LogiAuto mit jeder OLE-Automation fähigen Applikation aufrufen und mit Befehlen steuern.



Der Dialog ist in der Grösse veränderbar und weist folgende Elemente für die Bedienung auf:

- Menü
- Meldungsfield
- Statuszeile

## Bedienung über das Menü

Im Modul LogiAuto kann das Menü mit der Maus oder mit Tastenkombinationen (z.B. ALT-D -> Datei) bedient werden.



Menü Datei (ALT-D)

- Meldungsfeld löschen (ALT-M), löscht das Meldungsfeld
- Beenden (ALT-B, ALT-X), Beenden der Applikation

Menü Hilfe (ALT-H)

- Hilfe über Hilfe (ALT-H), die Hilfe über das Hilfesystem wird angezeigt
- Übersicht Applikation (ALT-S), die Übersicht über die Applikation wird angezeigt
- Informationen (ALT-I, Umschalt+F1), der Dialog mit Informationen über die Applikation wird angezeigt

## Anzeige im Meldungsfeld

Die Anzeige im Meldungsfeld gibt laufend den Zustand und den Stand der Abarbeitung an. Für jeden Datensatz wird das Meldungsfeld gelöscht. Scrollen Sie im Meldungsfeld zurück, wenn Sie die Abarbeitung kontrollieren möchten.

## Anzeigen in der Statuszeile

Die Statuszeile ist unterteilt in verschiedene Anzeigebereiche:



- Anzeige für Hilfetext
- Halter rechts für die Grössenänderung

**Programmierschnittstelle**

Nachfolgend ist die Schnittstelle für Programmierer für das Modul LogiAuto der Applikation WinLogi beschrieben:

**OLE-Referenz:** WinLogi.AppAuto

**Typbibliothek:** LogiAuto.tlb

**function GetVersion: WideString;**

Gibt die Version des Moduls LogiAuto als String zurück.

**function NewRecord: Integer;**

Erstellt einen neuen Datensatz für die Störung und gibt die Störungsnummer zurück.

**function GetRecord(ID: Integer; out sTime: WideString; out sDate: WideString;**

out sPriority: WideString; out sVisum: WideString; out sLocation: WideString;

out sRegion: WideString; out sDetail: WideString; out sAlarm: WideString) :

Integer;

Gibt die Störungsdaten des gewünschten Datensatzes mit der angegebenen Störungsnummer zurück.

**function DeleteRecord(ID: Integer): Integer;**

Löscht den gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeTime(ID: Integer; const sTime: WideString): Integer;**

Wechselt die Zeit der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeDate(ID: Integer; const sDate: WideString): Integer;**

Wechselt das Datum der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeVisa(ID: Integer; const sVisa: WideString): Integer;**

Wechselt das Visum des Benutzers der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangePriority(ID: Integer; const sPriority: WideString): Integer;**

Wechselt die Priorität der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeLocation(ID: Integer; const sLocation: WideString): Integer;**

Wechselt den Ort der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeRegion(ID: Integer; const sRegion: WideString): Integer;**

Wechselt die Region der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Die Region ist vom Ort abhängig. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeDetail(ID: Integer; const sDetail: WideString): Integer;**

Wechselt das Detail der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Das Detail ist von der Region abhängig. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function ChangeAlarm(ID: Integer; const sAlarm: WideString): Integer;**

Wechselt den Alarmtext der Störung beim gewünschten Datensatz mit der angegebenen Störungsnummer. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function GetPriorities(out sPriorities: WideString): Integer;**

Gibt eine Liste aller Prioritäten zurück, mit der höchsten Priorität am Anfang. Die Einträge sind durch Tabulatoren getrennt. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function GetLocations(out sLocations: WideString): Integer;**

Gibt eine Liste aller möglichen Orte zurück. Die Einträge sind durch Tabulatoren getrennt. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function GetRegions(const sLocation: WideString; out sRegions: WideString): Integer;**

Gibt eine Liste aller möglichen Regionen des angegebenen Orts zurück. Die Einträge sind durch Tabulatoren getrennt. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function GetDetails(const sRegion: WideString; out sDetails: WideString): Integer;**

Gibt eine Liste aller möglichen Details der angegebenen Region zurück. Die Einträge sind durch Tabulatoren getrennt. Gibt 0 zurück für Erfolg oder eine Fehlernummer.

**function GetErrorText(iError: Integer): WideString;**

Wandelt eine Störungsnummer in einen deutschen Text um.

**Voraussetzungen für die korrekte Ausführung**

- Die Borland Database Engine (BDE) ab Version 5.0 muss installiert sein.
- Es müssen die Einträge für die Applikation WinLogi in der Borland Database Engine (BDE) eröffnet sein. Sehen Sie dazu in der Installation der Applikation WinLogi nach.
- Eine gültige Seriennummer muss in der Registrierung eingetragen sein.
- Das Betriebssystem muss für die 2000 Jahr Problematik konfiguriert sein (siehe Beschreibung Y2000 Einstellungen).