ELOoffice - OLE Automation Schnittstelle

Wichtige Information:

Bitte beachten Sie den Charakter der OLE-Automation Schnittstelle:

1. Leichte und schnelle Erweiterbarkeit des ELO Clients um zusätzliche Funktionen
2. Möglichst hohe Aufwärtskompatibilität zwischen den Versionen

Diese beiden Hauptziele führen dazu, dass mitunter im Projekt in kurzer Zeit neue neue Automation Befehle eingeführt werden, die nur eine bestimmte Funktion ausführen sollen. Es ist also nicht sichergestellt, dass jeder Befehl in jeder Arbeitsansicht und in beliebiger Kombination mit anderen Befehlen ausführbar ist. Aufgrund der kompatibilitäts-Forderung können neuen Client-Funktionen nicht immer in die bestehenden Befehle eingefügt werden. In diesem Fall wird entweder eine xyz_EXT Funktion geschaffen oder die Funktionalität wird überhaupt nicht über die Automation Schnittstelle exportiert.

Bedenken Sie auch, dass es sich bei der Schnittstelle um eine Client-Schnittstelle handelt. Sie ist nicht dafür gedacht, dass über sie eine sehr große Anzahl von Aktionen durchgeführt wird. Wenn Sie hier in Grenzbereiche vordringen, sollten Sie vorher überprüfen, ob die OLE Schnittstelle Ihren Anforderungen überhaupt gerecht wird. Da hier viele Komponenten im Spiel sind auf die wir keinen direkten Einfluß haben, können wir bei manchen Beschränkungen (z.B. Memory Leaks in der OLE Schnittstelle, im Windows Scripting Host oder auch in der Compiler Laufzeitumgebung) keinen Work Around anbieten.

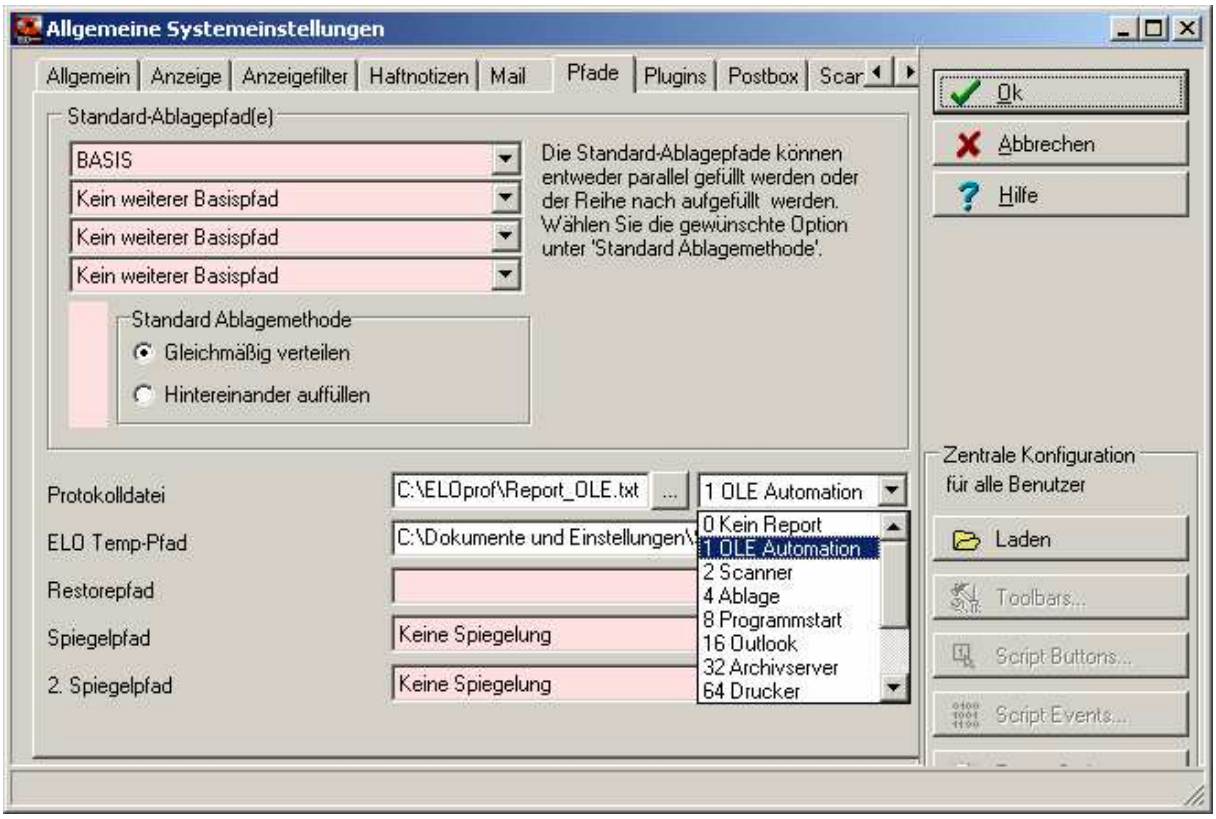
Die Beschreibung der ELOoffice OLE Automation Schnittstelle ist ein Auszug aus der ELOprofessional OLE Automation Dokumentation. Da die beiden Schnittstellen weitestgehend identisch sind, können viele Scripte mit geringem Aufwand sowohl unter ELOoffice wie auch unter ELOprofessional eingesetzt werden. In der ELOoffice Schnittstelle fehlen jedoch alle Aufrufe zu Funktionen, die nur unter ELOprofessional verfügbar sind (z.B. Workflow). Es kann aber vorkommen, dass in der Dokumentation vereinzelt ELOprofessional Funktionen übersehen wurden und die Scripte somit einen Fehler melden. Wenn Sie so einen Punkt finden, dann teilen Sie es uns bitte per EMail an support@elo.com mit, damit wir die Dokumentation korrigieren können.

Inhaltsverzeichnis

Erste Schritte

Die folgenden Abschnitte geben einen Überblick zu der Programmierung der ELO Automation Schnittstelle. Die Beispiele sind auf die notwendigen Schritte reduziert, es wird keine Fehlerkontrolle durchgeführt.

Ab der Version 3.0 haben Sie die Möglichkeit alle Automation Zugriffe in eine Datei zu protokollieren. Das ist für die Scriptentwicklung eine praktische Erweiterung. Ganz besonderes Gewicht erhält diese neue Funktion aber bei Fehlfunktionen oder der Kontrolle von fertigen Komponenten deren inneren Aufbau Sie nicht einsehen können. Eingeschaltet wird der Report über den Optionen-Dialog:



ELO starten, Archiv auswählen und Anmeldedialog

Für die ELO-Fernsteuerung gibt es prinzipiell zwei unterschiedliche Szenarien:

1. Ein Anwender arbeitet mit ELO und Sie wollen durch Ihre Applikation Aktionen im Arbeitsbereich dieses Anwenders durchführen. Es wird dann kein Login benötigt, es ist bereits durch den Anwender ausgeführt worden. Sie müssen in Ihrer Applikation dann nur prüfen, ob ELO bereits aktiv ist.
2. Sie haben einen eigenständigen Prozess (z.B. automatische Fax- oder Mail-Übernahme). In diesem Fall muß die Applikation ein Login (und am Ende ein Logout) durchführen.

Anmeldung an ELO für den Fall 1:

```
// ELO starten ...
try
  {
```

```
    EloServer= CreateOleObject("ELO.office");
  }
catch (...)
  {
    // ELOoffice muß mindestens einmal auf dem
    // Arbeitsplatz aufgerufen worden sein - es registriert
    // sich dann automatisch als OLE Automation Server
    return;
  };

// Zunächst sollten Sie sicherstellen, daß wirklich
// ein Anwender angemeldet ist.
ObjId=EloServer.OleFunction("GetEntryId",-1);
if (ObjId==-1)
  {
    // Fehler: kein Arbeitsbereich aktiv, d.h. es ist
    // zwar ELO gestartet aber kein Anwender angemeldet
    return;
  };
```

Anmeldung an ELO für den Fall 2:

```
// ELO starten ...
try
  {
    EloServer= CreateOleObject("ELO.office");
  }
catch (...)
  {
    // ELOoffice muß mindestens einmal auf dem
    // Arbeitsplatz aufgerufen worden sein - es registriert
    // sich dann automatisch als OLE Automation Server
    return;
  };

// Version kontrollieren
iOleResult=EloServer.OleFunction("Version");
if (iOleResult<101002)
  {
    // Version kleiner als 1.01.002
    // zu Alt!
    EloServer.OleFunction("Login","LOGOUT","","");
    EloServer=NULL;
    return;
  };

// Systemanmeldung
iOleResult=EloServer.OleFunction("Login", Loginname,
                                 Passwort, Archiv );
if (iOleResult<0)
  {
    // Unbekannter Loginname, Passwort oder Archiv
    EloServer.OleFunction("Login","LOGOUT","","");
    EloServer=NULL;
  };
```

Abmelden von ELO für den Fall 2:

```
// Abmelden vom Archiv und ELO wieder beenden
EloServer.OleFunction("Login","LOGOUT","","");
```

```
EloServer=NULL;
```

Verwenden des Trennzeichens "¶"

Für Pfadangaben wird das Trennzeichen "¶" (ALT 0182) verwendet. Früher war dieses das Zeichen "¿" (ALT 0191).
Um auch zukünftige Änderungen berücksichtigen zu können, wird dem Entwickler geraten, die Funktion "LookupDelimiter" zu verwenden. Diese Funktion liefert das aktuell gültige Trennzeichen zurück.

Ermitteln des aktuell ausgewählten Eintrags

Über die Funktion GetEntryId können Sie den aktuell selektierten Eintrag abfragen, GetEntryName liefert Ihnen den dazugehörenden Namen.

```
ObjectId=EloServer.OleFunction("GetEntryId",-1);
ObjectShort=EloServer.OleFunction("GetEntryName",ObjectId);
```

Liste aller Dokumente im Schrank Buchhaltung, Ordner Rechnung, Register 1998

Zuerst wird die ObjektId des Registers 1998 im Ordner Rechnungen im Schrank Buchhaltungen mit LookupIndex ermittelt. Nach einem Wechsel in das Register können alle Dokumente abgefragt werden.

```
// Ersmal das Register suchen
RegisterId=EloServer.OleFunction("LookupIndex",
                                 "¶Buchhaltung¶Rechnung¶1998");
if (RegisterId>0)
  {
    // nun das gefundene Register aufrufen
    EloServer.OleFunction ("GotoId",0-RegisterId);

    // ... und die Dokumente in dem Register abfragen
    for (i=0; i<10000; i++)
      {
        DocumentId=EloServer.OleFunction("GetEntryID",i);
        if (DocumentId<1) break;

        // hier wird das Dokument in eine Liste aufgenommen
      };
  };
```

Aufnahme eines neuen Dokuments in das Register

Zur Aufnahme eines neuen Dokuments muß zuerst ein leerer Eintrag vorbereitet werden, mit den Verschlagwortungsdaten gefüllt werden und die Imagedatei mit AddToPostbox in die Anwenderpostbox übertragen werden. Danach kann dieser Eintrag in das Archiv übernommen werden.

```
// leeren Datensatz vorbereiten und füllen
EloServer.OleFunction("PrepareObject",0,4,0);
EloServer.OlePropertySet("ObjShort","Urlaub in Florida");
EloServer.OlePropertySet("ObjMemo","Ford Home in Naples");
EloServer.OlePropertySet("ObjFlags",2);

// ab in die Postbox damit
iOleResult=EloServer.OleFunction("AddPostboxFile",
                                 "c:\temp\ford.tif");
```

```
// und aus der Postbox in das Archiv übertragen
sDestInfo="¶Urlaub 98¶Florida Westküste¶Ford Meyers";
iOleResult=EloServer.OleFunction("MoveToArchive",sDestInfo);
```

Masken-(Dokumententyp)-nummer aus der Bezeichnung ermitteln

Wenn Sie ein Dokument in das Archiv übertragen wollen, so müssen Sie zuerst den Dokumententyp festlegen. ELO arbeitet hier intern mit Maskennummern, die reale Welt mit Bezeichnungen. Zur Umsetzung der Bezeichnung in eine Maskennummer müssen Sie über die verfügbaren Masken laufen und mit der Bezeichnung vergleichen:

```
int __fastcall TMailImpMainDlg::FindEloMailMask()
  {
    AnsiString sTmp;
    int        iEloMask;

    iEloMask=0;
    for (iEloMask=0;iEloMask<MAX_MASKNO;iEloMask++)
      {
        if (EloServer.OleFunction("ReadObjMask",iEloMask)>=0)
          {
            sTmp=EloServer.OlePropertyGet("ObjMName");
            if (sTmp.UpperCase()=="ELOMAIL")
              {
                return iEloMask;
              };
          };
      };

    return -1;
  };
```

Einfacher geht es mit der Funktion LookupMaskName. Die oben beschriebene Funktion reduziert sich auf einen Aufruf:

```
iEloMask=EloServer.LookupMaskName("ELOMAIL");
```

Übertragen einer Mail mit Textkörper und Dateianbindung in das Archiv

Das folgende Beispiel zeigt wie eine e-Mail automatisch in das Archiv eingestellt wird. Hierzu wird diese Mail zuerst eingelesen, der Betreff, Absender und Empfänger ausgewertet und der Rumpf und die Dateianbindung wieder abgespeichert. An die Funktion InsertIntoELO werden dann die Basisinformationen und die Dateinamen übergeben.

```
bool __fastcall TMailImpMainDlg::InsertIntoElo( int iMailMask,
                                    const AnsiString sDestination,
                                    const AnsiString sSubject,
                                    const AnsiString sFrom,
                                    const AnsiString sTo,
                                    const AnsiString sMailText,
                                    const AnsiString sMailAttchment )
  {
    int iOleResult,iObjId;

    EloServer.OleFunction("PrepareObject",0,4,iMailMask);
    EloServer.OlePropertySet("ObjShort",sSubject);
```

```
    EloServer.OlePropertySet("ObjFlags",1);  // Versionskontrolliert
    if (iMailMask>0)
      {
        // die Maske EloMail besitzt zwei
        // Zusatzeinträge: Absender und Empfänger
        EloServer.OleFunction("SetObjAttrib",0,sFrom);
        EloServer.OleFunction("SetObjAttrib",1,sTo);
      };

    iOleResult=EloServer.OleFunction("AddPostboxFile",sMailText);

    // wenn kein Archivziel vorhanden ist bleibt
    // die Mail in der Postbox
    if (iOleResult>0 && sDestination!="")
      {
        iOleResult=EloServer.OleFunction("MoveToArchive",
                                         sDestination);

        // hier wird die neue ObjectId ermittelt ...
        iObjId=EloServer.OleFunction("GetEntryId",-2);
        if (iOleResult>0)
          {
            // ... und die Dateianbindung vorgenommen.
            iOleResult=EloServer.OleFunction("InsertDocAttachment",
                                             iObjId,sMailAttachment);
          };
      };

    return iOleResult>0;
  };
```

Bearbeiten einer Maske

Über die Befehle ReadObjMask und WriteObjMask können Sie Dokumenttypinformationen lesen und wieder speichern. Als **Dokumenttypinformation** stehen folgende Werte zur Verfügung:

ObjMName	Der Maskenname
ObjMIndex	Die Zielinformation bei automatischer Ablage
MaskFlags	Bit ...
MaskKey	Maskenschlüssel
DocKey	Vorgabewert für Dokumentenschlüssel
DocKind	Vorgabewert für Dokumentenfarbe
DocPath	Ablagepfad des Dokuments
DocTPath	Vorgabewert für den Ablagepfad

Zusätzlich stehen noch **50 Attributzeilen** (0..49) zur Verfügung. Jede Attributzeile besitzt folgende Werte:

Get/SetObjAttrib()	Anwendereingabe, z.B. eine Rechnungsnummer
Get/SetObjAttribName()	Bezeichnung des Feldes in der Maske, z.B. „Rechnungsnummer“
Get/SetObjAttribKey()	Indexbezeichnung in der Datenbank, z.B. „RENR“
Get/SetObjAttribType()	0: Text, 1: Datum, 2: Numerisch
Get/SetObjAttribMin()	Minimale Eingabelänge in Zeichen, 0: keine Kontrolle
Get/SetObjAttribMax()	Maximale Eingabelänge in Zeichen, 0: keine Kontrolle

Das Feld ObjAttrib ist bei der Definition einer Eingabemaske ohne Belang, es wird hier nicht mit abgespeichert, da es ja erst später bei der Anlage von Dokumenten die Anwendereingabe tragen wird.

Die Unterscheidung ObjAttribName und ObjAttribKey wird aus drei Gründen durchgeführt:

- Wenn unterschiedliche Sprachversionen zusammenarbeiten müssen, dann kann in den Eingabemasken ein geeigneter Text (z.B. „Rechnungsnummer“ und „Invoice No.“) aufgeführt werden. Datenbankintern wird jedoch in beiden Fällen der gleiche Schlüssel verwendet (z.B. „RENR“).
- Eine Maske kann mehrere gleichartige Felder enthalten. Z.B. kann eine Eingabemaske „Buch“ mehrere Zeilen zu den Autoren enthalten (Autor, Co-Autor etc). Alle diese Zeilen können dann Datenbankintern auf den gleichen Schlüssel gelegt werden (z.B. Autor) und deshalb gleichermaßen bei der Suche berücksichtigt werden.
- Unterschiedliche Masken können gleichartige Felder enthalten. Eine Rechnung enthält das Feld „Kundenname“, ein Lieferschein das Feld „Lieferantenname“. Beide Felder können den internen Datenbankschlüssel „Name“ erhalten und somit bei der Recherche direkt angesprochen werden.

Beachten Sie bitte, daß die minimale und maximale Eingabe sich auf die Anzahl der Zeichen und nicht (auch nicht bei numerischen Feldern) auf den Eingabewert.

Die Indexzeilen 0..49 stehen für die Verschlagwortungsfelder zur Verfügung. Danach folgen 10 Zeilen, die für ELO interne Aufgaben reserviert sind. Die Zeile 50 enthält im Augenblick die Link-Information. Hierzu erhält die Indexzeile die Gruppenbezeichnung ELO_XLINK im Feld AttribName und eine 12-stellige Zufallszeichenkette im Wert Feld.

Zum Anlegen einer neuen Maske hier ein kurzes Beispiel:

```
NEWMASK=9999
Set Elo=CreateObject("ELO.office")

if Elo.ReadObjMask( NEWMASK )<0 then
  MsgBox "Fehler beim Vorbereiten der Maske"
else
  Elo.ObjMName="ELO Testmaske"
  Elo.MaskFlags=25

  ' eine Indexzeile, Eingabelänge 5..10 Zeichen
  call Elo.SetObjAttribName( 0, "Index 1" )
  call Elo.SetObjAttribKey( 0, "IDX1" )
  call Elo.SetObjAttribMin( 0, 5 )
  call Elo.SetObjAttribMax( 0, 10 )

  x=Elo.WriteObjMask()
  if x<0 then
    MsgBox "Fehler Nr. " & x & " beim Anlegen der neuen Maske."
  else
    MsgBox "Neue Maske mit der Nummer " & Elo.ObjMaskNo & " angelegt."
  end if
end if
```

Einlesen eines Farbwertes oder der gesamten Farbtabelle

Zum Einlesen eines Farbwertes steht Ihnen die Funktion ReadColorInfo und die Properties ColorInfo und ColorName zur Verfügung.

```
iOleResult=EloServer.OleFunction("ReadColorInfo",MyColorNumber);
if (iOleResult>0)
  {
    MyColorRGB=EloServer.OlePropertyGet("ColorInfo");
    MyColorName=EloServer.OlePropertyGet("ColorName");
  };
```

Zum Ermitteln der kompletten Farbtabelle steht Ihnen eine Sonderform von ReadColorInfo zur Verfügung. Wenn Sie das Bit 0x8000 beim Lesen in der Farbnummer setzen, dann liest die Funktion die gewünschte Farbe oder falls sie nicht vorhanden ist, die nächst größere Farbnummer.

```
MyColorNumber=0;
for (;;)
  {
    MyColorNumber=MyColorNumber|0x8000;
    iOleResult= EloServer.OleFunction("ReadColorInfo",MyColorNumber);
    if (iOleResult<0) break;

    MyColorNumber= EloServer.OlePropertyGet("ColorNo");
    // Hier jetzt den aktuellen Farbwert bearbeiten

    MyColorNumber++;
  };
```

Abfrage und Einstellen der Arbeitsansicht

Zum Abfragen oder Einstellen der aktuellen Arbeitsansicht steht die Funktion SelectView zur Verfügung. Wenn Sie diese Funktion mit dem Parameter 0 aufrufen erhalten Sie als Rückgabewert die Nummer des aktuell eingestellen Arbeitsblattes (1..5). Wenn Sie einen der Werte 1..5 verwenden wird auf die gewählte Ansicht umgeschaltet.

```
// aktuellen Stand abfragen
ActView=EloServer.OleFunction("SelectView",0);

if (ActView!=3)
  {
    // ist nicht auf die Postbox eingestellt -> umschalten
    EloServer.OleFunction("SelectView",3);
  };
```

Postbox-Inhalt bearbeiten

Das folgende Beispiel zeigt, wie ein externes Programm oder ein ELO Scripting-Makro die Postbox durchsuchen und die vorhandenen Einträge bearbeiten kann.

In diesem Beispiel werden allen Einträge geladen und die selektierten Einträge im Memo-Feld durch den Text „Wichtig“ und die nicht selektierten Einträge durch den Text „Unwichtig“ ergänzt.

```
for (iPostLine=0;i<1000;i++)  // maximal 1000 Einträge beachten
  {
    // erstmal die gewünschte Zeile laden
    iRes=EloServer.OleFunction("PrepareObject",-1,iPostLine,0)
    switch (iRes)
      {
        case -5: continue;    // keine Verschlagwortung vorhanden
        case -7: continue;    // keine Verschlagwortung vorhanden
        case -6: return true; // fertig, keine weiteren Einträge
        case 3:  sText="Wichtig"; break;
        case 4:  sText="Unwichtig"; break;
        default: return false; // da ist was schiefgegangen
      };

    // Inhalt des Memofeldes lesen, Zusatztext anfügen und zurück
```

```
    sText=EloServer.OlePropertyGet("ObjMemo")+sText;
    EloServer.OlePropertySet("ObjMemo",sText);

    // geänderten Datensatz wieder speichern
    EloServer.OleFunction("AddPostboxFile","");
  };
```

Dokumente quer einscannen

Viele Einzugsscanner können DIN A4 Dokumente nur in Längsrichtung einscannen. Falls eine Reihe von Dokumenten quer eingelesen werden muß, ist für jedes einzelne Dokument ein Eingriff des Anwenders notwendig. Über ein einfaches Skript kann dieser Vorgang automatisiert werden. Legen Sie sich ein neues Skript mit folgendem Inhalt an:

```
Set Elo=CreateObject("ELO.office")
res=Elo.RotateFile("",90)
```

Nun melden Sie dieses Skript für das Ereignis „nach dem Scannen“ an. Es wird automatisch nach jeder eingelesenen Seite ein Drehvorgang ausgelöst.

Dieses Skript kann auch leicht so erweitert werden, daß es die komplette Postliste nach ausgewählten Einträgen durchsucht und diese dann dreht. In dieser Form kann es dann auf eine anwenderdefinierte Schaltfläche gelegt werden und so eine größere Gruppe von Dokumenten gleichzeitig gedreht werden.

Einträge über die OLE Schnittstelle suchen

Dieses Beispiel geht von folgenden Szenario aus:

> In einer Firma werden Lieferscheine mit Barcode ausgedruckt. Diese Lieferscheine werden dann im Lager mit handschriftlichen Notizen ergänzt und können deshalb nicht direkt per COLD übertragen werden. Statt dessen werden Sie wieder eingescannt und über die Barcode Komponente im Archiv abgelegt.
>
> Im Beispiel sollen dann regelmäßig aus der Auftragsverwaltung die ELO Dokumente überprüft werden (ist der Lieferschein mittlerweile eingescannt worden?) und mit zusätzlichen Informationen gefüllt werden.
>
> Die Ablagemaske für die Lieferscheine enthält eine Zeile „Lieferscheinnummer – LFNR“ welche über die Barcodekomponente gefüllt wird und die Zeilen „Kundennummer – KDNR“ und „Lieferdatum – LFDAT“ welche dann aus der Auftragsverwaltung ergänzt werden.

Als Vorbereitung für die nachfolgenden Zeilen muß die Applikation ein EloServer Objekt erzeugen und aus der Liste der Masken diejenige mit dem Namen „Lieferscheine“ ermitteln (iLfMaskNo).

Zuerst muß die Auftragsverwaltung also alle noch offenen Lieferscheine durchgehen – für jeden Lieferschein wird dann die folgende Routine aufgerufen (beachten Sie bitte, daß eine „echte“ Routine entsprechende Fehlerkontrollen enthalten sollte, welche hier der Klarheit halber entfernt wurden“):

```
bool AddInfo( AnsiString sLfnr, AnsiString sKdnr, AnsiString sLfdat )
  {
    EloServer.OleFunction("PrepareObject",0,0,iLfMaskNo);
    EloServer.OleFunction("SetObjAttrib",0,sLfnr);
    iCnt=EloServer.OleFunction("DoSearch");
    switch (iCnt )
      {
        case -1: // Fehler
          return false;
```

```
            case 0:  // nicht gefunden
              return false;
            case 1:  // gefunden - nun Eintragen
              break;
            default: // Mehrfache Einträge, das muß irgendwie behandelt
                     // werden, z.B. alle Einträge werden ergänzt.
              return false;
          };

        iObjId=EloServer.OleFunction("GetEntryId",0)
        EloServer.OleFunction("PrepareObject",iObjId,4,iLfMaskNo);
        EloServer.OleFunction("SetObjAttrib",1,sKdnr);
        EloServer.OlePropertySet("ObjXDate",sLfdat);
        EloServer.OleFunction("UpdateObject");
        return true;
      }
```

Als Folge des Funktionsaufrufs muß die Auftragsverwaltung dann nur noch eine Fehlermeldung ausgeben oder ihre interne Datenbank um die Information ergänzen, daß der Abgleich erfolgreich durchgeführt werden konnte. Diese Vorgehensweise bietet den Vorteil, daß eine Kontrolle gegeben ist, ob alle Lieferscheine wieder eingescannt worden sind, und es muß keine weitere manuelle Erfassung der in der EDV ohnehin vorhandenen Daten (Kundennummer, Datum) erfolgen.

EloScript-Beispiel: Zugriffspfad auf das aktuell selektierte Objekt ausgeben

```
Set Elo=CreateObject("ELO.office")

STemp=""
id=Elo.GetEntryId(-1)
ires=Elo.PrepareObject(id,0,0)
STemp=Elo.ObjShort

while ires=2 and Elo.ObjMainParent>1
  ires=Elo.PrepareObject(Elo.ObjMainParent,0,0)
  STemp=Elo.ObjShort & " : " & STemp
wend

MsgBox "Zugriffspfad: "&STemp
```

EloScript-Beispiel: Leer- und Trennseitenüberprüfung in der Postbox

```
' TestPage.VBS  10.10.2001
'-----------------------------------------------------------------
' © 2001 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo-digital.de)
'-----------------------------------------------------------------
' Dieses Skript überprüft alle Einträge der Postbox auf Leer-
' oder Trennseiten. Falls eine Leerseite (mindestens 97% Weißanteil
' in jedem Segment) gefunden wurde, wird zudem noch geprüft, wie
' "leer" sie ist und der entsprechende Wert ausgegeben.
' -----------------------------------------------------------------

set Elo = CreateObject("ELO.office")
ver=Elo.Version
if ver<600220 then
  MsgBox "Dieses Skript benötigt mindestens die Version ELO 6.00.220"
```

```
else
  ' Über alle Postboxeinträge laufen, maximal jedoch 100
  for i=0 to 100
    ' Eintrag in den Viewer auswählen
    if Elo.SelectLine(i)<0 then
      exit for
    end if

    ' Prüft auf Trenn- und Leerseite gleichzeitig
    res=Elo.CheckPage(3, 970)
    if res=1 or res=3 then
      ' ist eine Leerseite, nun den Grenzwert ermitteln
      for j=970 to 999
        if Elo.CheckPage(1,j)=0 then
          exit for
        end if
      next
      sTmp=sTmp & " [ IsWhite: " & j/10.0 & " % ] "
    end if

    ' Trennseitentext anfügen
    if res=2 or res=3 then sTmp=sTmp & " [ IsTrennseite ] "

    ' Dateiname in die Meldung aufnehmen
    sPathName=Elo.ActivePostFile
    sName=Elo.SplitFileName( sPathName, 2 )
    sTmp=sTmp & " : " &sName & vbcrlf

    ' führt intern ein ProcessMessages aus, sonst nichts
    Elo.UpdatePostboxEx 23,1

    ' Nimmt das SelectLine zurück
    Elo.UnselectPostboxLine(i)
  next

  ' Ergebnis in einer MessageBox anzeigen.
  MsgBox sTmp
End if
```

Eine einfache Formularverwaltung

Das folgende Beispiel durchläuft die Postliste, nimmt jeden ausgewählten Eintrag und durchsucht mittels der OCR Software einen definierten Bereich nach den Stichwörtern „Rechnung“ und „Lieferschein“. Wenn eines dieser Stichwörter gefunden wird, liest es die dazu passende Maskendefinition ein und durchsucht weitere Bereiche des Formulars nach Rechnungsnummer, Kundennummer etc. Diese Einträge werden dann mit dem Dokument gespeichert und im Archiv abgelegt.

```
' Das Programm liest alle selektierten Einträge der Postbox und
' prüft, ob das Dokument zum Maskentyp "Rechnung" oder "Lieferschein"
' gehört. Dann werden die entsprechenden Daten der Maskzeilen aus
' dem Formular gelesen und gespeichert.
' Anschliessend wird das Dokument in das Archiv verschoben.



set ELO = CreateObject("ELO.office")
iNum=0
Do while (1)
```

```
  iRet=Elo.PrepareObject (-1,iNum,0)
  If (iRet = -6) Then
    MsgBox "Programm beendet!"
    Exit Do
  End If


  if (iRet = 3 Or iRet = -7) Then   ' Nur selektierte Dokumente bearbeiten.
    SaveObjShort=Elo.ObjShort
    ELo.ObjShort=""
    iRet=Elo.AddPostBoxFile("")
    strZeile="#"+CStr(iNum)
    res=Elo.AnalyzeFile(strZeile,"R(333,100,666,200)P(1)S(SHORT=1,100)",0)
    If (res = 1) Then
      obs=Trim(Elo.ObjShort)
      i=1
      do while i<= len(obs)
        if mid(obs,i,1)=" " then
          obs=mid(obs,1,i-1)+mid(obs,i+1,Len(obs)-i)
        end if
        i=i+1
      loop
      If (InStr(obs,"Rechnung")>0) then
        res=Elo.Status( "Rechnung in Zeile " & strZeile & " erkannt.")
        res=Elo.AnalyzeFile(strZeile,"R(600,100,1000,200) P(1) S(ReNr=1,10)
R(450,180,1000,220) P(1) S(KdNr=1,10) R(450,220,1000,260) P(1) S(Ort=1,10)
",6)
        If (res = 1) Then
          RechNr=Elo.GetObjAttrib(0)
          iRet=Elo.PrepareObject (-1,iNum,0)
          ELo.ObjShort=ELo.ObjShort+RechNr
          iRet=Elo.AddPostBoxFile("")
          ObjIndex="REG=Rechnung"
          iRet=Elo.MoveToArchive (ObjIndex)
          if (iRet <> 1) then
            msgbox "Fehler beim Übertragen, RetValue = " & Cstr(iRet)
          End If
        End If
      Else
        If (InStr(Elo.ObjShort,"Lieferschein")>0) then
          res=Elo.Status( "Liefersch. in Zeile " & strZeile & " erkannt.")
          res=Elo.AnalyzeFile(strZeile,"R(600,100,1000,200) P(1)
S(LfNr=1,10) R(500,180,1000,220) P(1) S(KdNr=1,10) R(500,220,1000,260) P(1)
S(Produkt=1,10) R(500,260,1000,320) P(1) S(LagerOrt=1,50)  ",7)
          LieferNr=Elo.GetObjAttrib(0)
          iRet=Elo.PrepareObject (-1,iNum,0)
          ELo.ObjShort=ELo.ObjShort+LieferNr
          iRet=Elo.AddPostBoxFile("")
          ObjIndex="REG=Lieferschein"
          iRet=Elo.MoveToArchive (ObjIndex)
          if (iRet <> 1) then
            msgbox "Fehler beim Übertragen, RetValue = " & Cstr(iRet)
          End If
        Else
          res=Elo.Status( "Unbekanntes Dokument in " & strZeile )
          If SaveObjShort="" then
            Elo.ObjShort="Unbekanntes Dokument"
          Else
            Elo.ObjShort=SaveObjShort
          End If
          iRet=Elo.AddPostBoxFile("")
        End If
      End If
    Else
```

```
      res=Elo.Status( "Dokumententyp konnte nicht geprüft werden in Zeile "
& strZeile )
      If SaveObjShort="" then
        Elo.ObjShort="Unbekanntes Dokument"
      Else
        Elo.ObjShort=SaveObjShort
      End If
      iRet=Elo.AddPostBoxFile("")
    End If
  End If
iNum=iNum+1
Loop

res=Elo.UpdatePostbox()
```

Beispiel: Vor dem Bearbeiten einer Haftnotiz

Beim Bearbeiten einer Haftnotiz wird automatisch der Name und die Uhrzeit in das Textfeld eingetragen. Der Parameter ScriptActionKey enthält dabei die Werte 1: vor dem Aufruf, 2: nach dem Aufruf, mit Ok abgeschlossen und 3: nach dem Aufruf, mit Abbruch abgeschlossen.

```
set Elo=CreateObject("ELO.office")
if Elo.ScriptActionKey=1 then
  note=Elo.NoteText
  if note<>"" then
    note=note & vbcrlf & vbcrlf & "====" & vbcrlf
  end if
  Elo.ReadUser( Elo.ActiveUserId )
  note=note & Date & Time & ": " & Elo.UserName & vbcrlf & "----" & vbcrlf
  Elo.NoteText=note
end if
```

Beispiel eines Microsoft Winword 8 Makros

Die folgenden Zeilen enthalten ein Beispiel für die automatische Übertragung eines Dokuments aus WinWord in ELO per Makro.

```
Public Sub MAIN()
TempVerz$ = Environ("TEMP")          ' Temoporäres Verzeichnis aus Umgebungsvariablen
If Documents.Count < 1 Then          ' Wenn kein Dokument offen
    MsgBox ("Übertragung ist nicht möglich, da kein Dokument geöffnet wurde")
    GoTo ENDE
End If
If Tasks.Exists("Der Elektronische Leitz Ordner") = False Then
    ' wenn ELO  nicht aktiv ist ->
    GoTo EloProgrammStarten          ' Meldung, daß ELO gestartet sein muß
End If
i = 1                                 ' Möglichen Temp-Dateinamen "WW_ELOi.doc"
Schleife:                             ' ermitteln, der nicht in der Fensterliste existiert
For j = 1 To Application.Windows.Count      '  Untersuche für alle geöffneten Fenster
    If Application.Windows(j).Caption="WW_ELO"+LTrim(Str(i))+".doc" Then
       i = i + 1                      '  nächsten temporären Dateinamen "WW_ELOi.doc"
       GoTo Schleife                  '  in Fensterliste suchen
    End If
Next j
TempName$ = "WW_ELO" + LTrim(Str(i)) + ".doc"
If Dir(ActiveDocument.FullName) <> "" Then  ' Wurde das Dokument schon mal gespeichert ?
   If ActiveDocument.Saved = False Then     ' Ist gespeichert, wurde aber verändert
     MsgText$ = "Das Dokument wurde verändert und bekommt den temporären Dateinamen: " +
        TempName$       + Chr$(13) + Chr$(10)
        Dateiname$ = TempVerz$ + "\" + TempName$
        ActiveDocument.SaveAs FileName:=Dateiname$
   Else                                ' ist gespeichert und wurde nicht verändert
     MsgText$ = ""                     ' keinen Hinweis, daß Dokument nur temporär existiert
        Dateiname$ = ActiveDocument.FullName ' Dateiname+Pfad (übernimmt die Orginal-Datei)
```

```
   End If
Else                                ' Dokument ist ein neues Dokument
   MsgText$ = "Achtung: Das Dokument ist noch nicht gesichert und bekommt den temporären
   Dateinamen: " +  TempName$ + Chr$(13) + Chr$(10)
   Dateiname$ = TempVerz$ + "\" + TempName$     ' temporären Dateinamen festlegen
   ActiveDocument.SaveAs FileName:=Dateiname$  ' abspeichern unter temporärem Dateinamen
End If
KurzName = ActiveDocument.BuiltInDocumentProperties("Title")
If KurzName = "" Then               ' falls kein Titel eingetragen ist, bekommt es
  KurzName = ActiveDocument.Name     ' den Dokumenten-Namen als Kuzrbezeichnung
End If
Kom = ActiveDocument.BuiltInDocumentProperties("Comments")    ' Übernahme des Kommentars
DDate$ = ActiveDocument.BuiltInDocumentProperties("Last Save Time")
DDate$ = DateValue(DDate$)          ' Ermittelt gültiges Datum: tt.mm.jj
Dim ELOServer As Object
Set ELOServer = CreateObject("ELO.office")   ' ELO32 OLE-Server
On Error GoTo ANMELDEN                       ' ELO ist im AnmeldeDialog
iOleResult = ELOServer.PrepareObject(0, 4, 0)     ' neuer Eintrag, Dokument, Standard
On Error GoTo 0
ELOServer.ObjMemo = Kom             ' Dokumenten-Kommentar
ELOServer.ObjXDate = DDate$         ' letztes Speicherdatum
EntryID = ELOServer.GetEntryID(-12)  ' >0 -> Es existiert ein aktiviertes Register
If (ELOServer.SelectView(0) = 1) And (EntryID > 0) Then ' ELO steht in "Archiv + Register"
   txt = MsgText$ + Chr$(13) + Chr$(10) + "Das Dokument wird unter folgender Kurzbezeichnung
   im Archiv abgelegt."
   KurzName = InputBox(txt, "ELOoffice Dokumentenübergabe", KurzName)
                                    ' Kurzname kann mittels
   ELOServer.ObjShort = KurzName        ' Dialog verändert werden
   iOleResult = ELOServer.AddPostboxFile(Dateiname$)              ' zuerst in die Postbox,
   iOleResult = ELOServer.MoveToArchive("#" + Str(EntryID))       ' dann ins Archiv
Else                                ' sonst Übertragung in die Postbox
   txt = MsgText$ + Chr$(13) + Chr$(10) + "Das Dokument wird unter folgender Kurzbezeichnung
   in die Postbox abgelegt."
   ELOServer.ObjShort = InputBox(txt, "ELOoffice Dokumentenübergabe", KurzName)
   iOleResult = ELOServer.AddPostboxFile(Dateiname$)              ' Übergabe in die Postbox
End If
Set ELOServer = Nothing                 ' Server-Object wieder freigeben
GoTo ENDE

ANMELDEN:
Set ELOServer = Nothing                 ' Server-Object wieder freigeben
MsgBox ("Fehler: Sie müssen sich ELOoffice anmelden.")
GoTo ENDE

EloProgrammStarten:
MsgBox ("Fehler: ELOoffice muß zur Übertragung aktiv sein.")

ENDE:
End Sub
```

Scriptereignis "Eintragen/Verschieben einer Objektreferenz"

Über das Scriptereignis "Beim Eintragen/Verschieben einer Objektreferenz" können Sie darauf reagieren, wenn ein Anwender ein Dokument oder ein Ablagestrukturelement im Archiv verschiebt. Hier könnte man z.B. Anpassungen in der Berechtigungsstruktur oder der Verschlagwortung vornehmen.

```
Set Elo=CreateObject( "ELO.office" )

ObjectId=Elo.NodeAction
ParentId=Elo.ScriptActionKey
NewParent=Elo.WvNew

if Elo.ActionKey=1 then
  MsgBox "AddRef  ObjId=" & ObjectId & "  Parent: " & ParentId
End if
if Elo.ActionKey=2 then
  MsgBox "CopyRef  ObjId=" & ObjectId & "  OldParent: " & _
         ParentId & "  NewParent: " & NewParent
end if
if Elo.ActionKey=3 then
```

```
  MsgBox "MoveRef  ObjId=" & ObjectId & "  OldParent: " & _
         ParentId & "  NewParent: " & NewParent
end if
```

Über den ActionKey können Sie erkennen, ob ein Objekt neu eingefügt wird (1), kopiert (2) oder verschoben wird (3). Die weiteren Parameter wie ObjektId und Vorgänger können über die OLE Schnittstelle ermittelt werden. Dieser Aufruf wird nur aktiviert, wenn das Verschieben/Einfügen direkt über den Client durchgeführt wird. Falls die Operation über die OLE Schnittstelle aktiviert wurde, wird dieses ScriptEvent nicht getriggert.

Scriptereignis "Beim Aus/Einchecken eines Dokuments"

Über dieses Scriptereignis erhalten Sie zu unterschiedlichen Zeiten Benachtrichtigungen über ein- oder auszucheckende Dokumente. Die einzelnen Zeitpunkte werden über den ActionKey spezifiziert, folgende Werte stehen dabei zur Verfügung :

Wert	Aktion
30	Nach dem Auschecken eines Dokuments aber vor der Aktivierung der Applikation zum Bearbeiten. Das Property CheckInOutFileName enthält den Namen der Datei. Über das Property ScriptActionKey können Sie erfahren, ob ein Dokument direkt in der Archivansicht aus einer Dokumentenvorlage entstanden ist, in diesem Fall ist das Bit 8 (Wert = 256) auf 1 gesetzt. Weiterhin enthält dieses Property die Information, ob das Dokument anschließend per ShellExecute aktiviert werden soll. 0: nicht aktivieren, 1: aktivieren ohne Nachfrage, 2: aktivieren mit Nachfrage. Sie können diesen Wert im Script auch verändern und somit die geplante Anzeigeart verändern.
31	Unmittelbar vor dem Einchecken eines Dokuments. Das Property CheckInOutFileName enthält den Namen der einzucheckenden Datei.
32	Nach dem Einchecken des Dokuments. Das Property CheckInOutFileName enthält den Namen der einzucheckenden Datei, diese wird unmittelbar nach diesem Event gelöscht werden.
33	Vor dem Auschecken eines Dokuments. Das Property CheckInOutObjID enthält die ELO ObjektId des auszucheckenden Dokuments. Über das Property ScriptActionKey können Sie bestimmen, ob ELO nach dem Scriptaufruf mit der normalen Bearbeitung fortfährt. Wenn Sie das Property unverändert auf –1 belassen, wird der Vorgang fortgesetzt. Wenn Sie den Wert 14 eintragen, dann geht ELO davon aus, dass Sie den CheckOut Vorgang komplett durch das Script abgearbeitet haben und keine weiteren Aktionen notwendig sind. Alle anderen Werte brechen die Bearbeitung ebenfalls ab und lösen eine entsprechende MessageBox aus.
34	Vor dem Einchecken eines Dokuments. Das Property CheckInOutFileName enthält den Namen der einzucheckenden Datei. Über das Property ScriptActionKey können Sie bestimmen, ob ELO nach dem Scriptaufruf mit der normalen Bearbeitung fortfährt. Wenn Sie das Property unverändert auf –1 belassen, wird der Vorgang fortgesetzt. Wenn Sie den Wert 14 eintragen, dann geht ELO davon aus, dass Sie den CheckOut Vorgang komplett durch das Script abgearbeitet haben und keine weiteren Aktionen notwendig sind. Alle anderen Werte brechen die Bearbeitung ebenfalls ab und lösen eine entsprechende MessageBox aus.
1001	Vor der Anzeige der Dateiliste für ein Register-CheckOut oder –CheckIn Vorgang. Dieses Event wird für jede Datei aufgerufen, der Dateiname steht im Property CheckInOutFileName, das Property CheckInOutObjId enthält die ELO ObjektId. Das Property ScriptActionKey enthält die Information, wie der Vorgang geplant ist (untersten 8 Bit), ob ein CheckIn (0x200) oder ein CheckOut (0x100) Vorgang aktiv ist. Zudem ist das Bit 0x10000000 gesetzt. Wenn dieses Bit vom

	ScriptEvent auf 0 gesetzt wird, dann wird diese Datei nicht in die Anzeigeliste aufgenommen. Achtung: dieses ScriptEvent wird bei einem Vorgang evtl. mehrfach aufgerufen – z.B. wenn der Anwender im Dialog eine Einstellung ändert und die Anzeigeliste deshalb neu aufgebaut wird.
1002	Vor dem CheckOut eines Dokument aus der Registerliste. Die weiteren Parameter werden wie beim Event 1001 gesetzt. Wenn das ScriptActionKey Bit 0x10000000 auf 0 zurückgesetzt wird, dann wird der CheckOut Vorgang für dieses Dokument unterdrückt.
1003	Nach dem CheckOut eines Dokuments aus der Registerliste.
1004	Vor dem CheckIn eines Dokuments aus der Registerliste. Die weiteren Parameter werden wie beim Event 1001 gesetzt. Wenn das ScriptActionKey Bit 0x10000000 auf 0 zurückgesetzt wird, dann wird der CheckIn Vorgang für dieses Dokument unterdrückt.
1005	Nach dem CheckIn eines Dokuments aus der Registerliste

Beispiel:

```
SET Elo=CreateObject( "ELO.office" )
if Elo.ActionKey=33 then
  Id=Elo.CheckInOutObjID
  Ext=UCase(Elo.GetDocExt( Id, 1 ))
  if Ext="MSG" then
    MsgBox "E-Mails können nicht ausgecheckt werden"
    Elo.ScriptActionKey=14
  end if
end if

if Elo.ActionKey=34 then
  File=Elo.CheckInOutFileName
  Ext=UCase(Right(File,3))
  if Ext="TIF" then
    MsgBox "Tiffs können nicht mehr eingecheckt werden."
    Elo.ScriptActionKey=14
  end if
end if
```

Scriptereignis Beim Bearbeiten der Verschlagwortung

Dieses Scriptereignis ist enthält eine Sammlung von Aktionen, die über den AktionKey unterschieden werden. Beim Start des Verschlagwortungsdialogs wird zuerst abgefragt, welche Indexzeilen einen anwenderdefinierten Button (24) erhalten sollen. Anschließend kommt die Mitteilung zum Start der Verschlagwortung (20). Jedes Betreten oder Verlassen einer Indexzeile löst ebenfalls eine Benachrichtigung aus, ebenfalls der Wechsel des Dokumententyps. Zur Beendigung der Verschlagwortung wird dann auch noch mal ein Ereignis ausgelöst.

Damit eigene Aktionsschaltflächen am Ende einer Indexzeile eingeblendet werden, muss das Event 24 bedient werden. Dieses erwartet als Rückgabe einen Vektor mit der Liste aller Buttons im Property TextParam. Dieser Vektor kann aus bis zu 54 '0' oder '1' Werten bestehen, jeder Eintrag ist für eine Indexzeile zuständig. Das erste Zeichen ist für die Kurzbezeichnung, das zweite für die Volltexteingabe im Suchdialog, die folgenden 2 sind für Erweiterungen reserviert und die letzten 50 sind für die 50 Indexzeilen. Wenn Sie also einen Button auf der Kurzbezeichnung und auf der 2. und 4. Indexzeile benötigen, dann müssen Sie den Wert 10000101 in TextParam eintragen (nur bis zur letzten 1 notwendig, die ganzen Nullen am Ende können weggelassen werden).

Wenn ein Anwender so eine Schaltfläche aktiviert, dann erhalten Sie in diesem Event einen AktionKey Wert von 3xxx, die erste Indexzeile 3000, die nächste 3001. Die Kurzbezeichnung liefert einen Wert 3999. Beachten Sie den zusätzlichen Offset, der eingetragen wird, wenn die Maske in der Suchansicht aufgerufen wird.

Im Normalfall werden die Schaltflächen in Abhängigkeit von der aktiven Dokumentenmaske unterschiedlich verwendet werden. Aus diesem Grund kommt die Abfrage nach dem Anzeigevektor nicht nur beim Start der Anzeige sondern zusätzlich noch bei jedem Wechsel des Dokumententyps.

Scriptereignis "Vor der Recherche"

Das Scriptereignis "Vor der Recherche" erlaubt die Veränderung der SQL Suchanfrage bevor sie an den SQL Server übergeben wird. Das SQL Kommando wird dabei im Property "TextParam" übergeben und Veränderungen durch das Script werden von dort aus von ELO übernommen.

Beachten Sie bitte, dass dieses Script bei jeder Suche aktiviert wird, also auch bei internen Recherchen (z.B. für die Link-Liste). Sie müssen also vor einer Veränderung der Suchanfrage kontrollieren, ob die gewünschte Anfrage aktiv ist.

Scriptereignis "Beim Viewer Export"

Über das Scriptereignis "Beim Viewer Export können Sie an mehreren Stellen in den Exportvorgang eingreifen. Die unterschiedlichen Zeitpunkte können Sie anhand des ActionKey Properties erkennen. Das Zielverzeichnis für den Viewer können Sie im Property ActivePostFile auslesen. Umgekehrt können Sie über das Property ScriptActionKey einen vorzeitigen Abbruch der Aktion erreichen. Solange Sie diesen Wert unverändert auf 1 belassen, wird der Vorgang fortgesetzt, wenn Sie dort über Ihr Script eine 0 eintragen, wird abgebrochen.

Der "Zeitpunkt" 2 ist günstig für das Kopieren der Stichwortlisten. Unter ELOprofessional 3.0 können Sie hier die Befehle zum kopieren der gewünschten Stichwortlisten ausführen. Unter 4.0 kann das auch vom Anwender manuell konfiguriert werden. Allerdings können Sie auch hier durch entsprechende Scripteinstellungen den Wunsch des Anwenders durch eine eigene Kontrolle ersetzen.

ActionKey	Zeitpunkt
1	Vor dem Kopiervorgang der Viewerdaten. Wenn Sie hier abbrechen, dann haben Sie den gleichen Status als wäre der Viewer überhaupt nicht ausgewählt worden.
2	Nach dem Kopieren der Viewerdaten und vor dem Starten des Viewerimports. Wenn Sie hier abbrechen, dann haben Sie zwar die allgemeinen Viewerdateien, Ihr Exportdatensatz wird jedoch nicht eingelesen.
3	Nach dem Start des Viewers. Beachten Sie bitte, dass zu diesem Zeitpunkt der eigentliche Importvorgang im Viewer noch nicht abgeschlossen ist. Leider gibt es keinen einfachen Weg, diesen Zeitpunkt per Script festzustellen.
4	(Ab ELOprofessional 4.0) Vor dem Kopieren einer Stichwortliste, den Dateinamen und Pfad finden Sie im Property ActivePostFile. Wenn Sie den ScriptActionKey auf 0 stellen, dann wird diese Datei nicht mit kopiert. Hierüber können Sie (falls der Anwender das Kopieren der Stichwortlisten aktiviert hat) steuern, welche Listen tatsächlich mitkopiert werden.

Beispiel:

```
Set Elo=CreateObject("ELO.office")

MsgBox Elo.ActionKey & " : " & Elo.ActivePostFile
```

Scriptereignis “Beim Stapelscannen”

Das Scriptereignis „Beim Stapelscannen“ wird von unterschiedlichen Stellen aus aktiviert. Hierüber kann auf die ungeklammerten Dokumente, auf die geklammerten Dokumente sowie auf den eigentlichen Ablagevorgang

Einfluß genommen werden. Die Dokumentenmaske für die Stapelablage wird über die Postboxfunktion „Ablagemaske voreinstellen“ (wie bei der Barcode-Ablage) bestimmt. Hierüber wird auch das normale Ablageziel bestimmt, die Maske muss also einen Ablageindex besitzen.

AktionKey=1: Beim Ablegen ins Archiv

Dieser Aufruf findet für jedes (bereits geklammerte) Dokument unmittelbar vor dem Ablegen statt. Das Property ActivePostFile zeigt auf die Scandatei, MainParentId auf das geplante Ziel, die anderen verschlagwortungsbezogenen Properties (z.B. Kurzbezeichnung, Indexzeilen) sind ebenfalls bereits geladen. Über das Script können an dieser Stelle noch beliebige Änderungen an der Verschlagwortung vorgenommen werden. Zusätzlich kann das Ziel verändert werden.
Die Möglichkeiten dieses Aufrufs schließen auch eine komplett scriptgesteuerte Ablage ein. In diesem Fall ist das Script für den eigentlichen Ablagevorgang und für das Löschen der Daten- und Verschlagwortungsdatei verantwortlich. Diesen Fall signalisiert der Client durch setzen des Property ScriptActionKey auf einen Wert ungleich Null.

AktionKey=2: Bei der Vorverarbeitung

Nach dem Scanvorgang wird automtisch eine Barcodeanalyse durchgeführt (wenn für die Dokumentenmaske ein Barcode definiert ist). Als nächstes wird eine Entscheidung über Start- und Folgeseiten der Dokumente getroffen. Per default ist jede Seite mit Barcode eine Startseite, alle anderen Seiten eine Folgeseite. Der Scriptaufruf wird dabei für alle Seiten ausgeführt. Durch setzen des Property DocKind (60: Startseite, ungelesen, 61: Folgeseite, ungelesen, 62: Startseite, gelesen, 63: Folgeseite, gelesen) kann das Script hier eine eigene Aufteilung vornehmen.

AktionKey=3: Vor dem Klammern

Dieser Aufruf wird vor dem Klammern gesendet. Dieses findet statt, wenn der Anwender die Dokumente per „A“ oder „S“ ins Archiv übertragen will.

AktionKey=4: Vor dem Ablegen ins Archiv

Nach dem Klammern erhält das Script über diesen Aufruf nochmals eine Benachrichtigung bevor die Übertragung ins Archiv begonnen hat. Hier können Aktionen durchgeführt werden, die den kompletten Stapel betreffen.

AktionKey=5: Nach dem Ablegen ins Archiv

Zum Abschluß der Aktion kann über diesen Aufruf noch eine Endeaktion, wie z.B. das Schreiben eines Reports stattfinden.

Beispiel:

Das folgende Beispiel setzt keine Barcode-Komponente vorraus, die Dokumentenanalyse findet statt dessen über einen OCR Vorgang statt (nur mit Volltext-Option möglich). Demodokumente für das Script können Sie leicht erzeugen, indem Sie die Seiten 100..200 aus diesem Dokument ausdrucken. Anhand der Wörter „Property“ oder „Funktion“ in Dokumentenkopf erkennt das Script die Startseiten, alle anderen Seiten werden als Folgeseiten deklariert. Als Ablagemaske sollten Sie ein Maske auswählen, die keinen Barcodeeintrag besitzt. Der Index mit dem Ablageziel muss auf jeden Fall eingetragen sein.

```
Set Elo=CreateObject("ELO.office")

' bei der Ablage passiert nichts, es wird das normale
' Ziel aus der Maskendefinition verwendet
if Elo.ActionKey=1 then
  ' in der Statuszeile das aktuell bearbeitete Dokument anzeigen
  Elo.Status Elo.ObjShort
```

```
end if

' da die Beispieldokumente keinen Barcode enthalten,
' muss die Einteilung Startseite/Folgeseite komplett
' vom Script übernommen werden.
if Elo.ActionKey=2 then
  Elo.Status "OCR-Verarbeitung " & Elo.ActivePostFile
  ' Dokumentenkopf per OCR auswerten
  call ELO.OcrClearRect()
  call ELO.OcrAddRect("100,80,999,200")
  call ELO.OcrAnalyze(Elo.ActivePostFile,0)
  'MsgBox Elo.OcrGetText(0)

  ' Wenn im Kopf der Text "Funktion", gefolgt von einem
  ' Funktionsnamen enthalten ist, dann ist es eine Startseite
  Cnt=Elo.OcrPattern(10,"*'Funktion'_*L*", Elo.OcrGetText(0))
  if Cnt>0 then
    Elo.ObjShort=Elo.OcrGetPattern(3)
    Elo.DocKind=60
  else
    ' Oder wenn im Kopf der Text "Property", gefolgt von einem
    ' Namen enthalten ist, dann ist es auch eine Startseite
    Cnt=Elo.OcrPattern(10,"*'Property'_*L*", Elo.OcrGetText(0))
    if Cnt>0 then
      Elo.ObjShort=Elo.OcrGetPattern(3)
      Elo.DocKind=60
    else
      ' Alles andere sind Folgeseiten
      Elo.ObjShort="Folgeseite"
      Elo.ObjMemo=Elo.OcrGetText(0)
      Elo.DocKind=61
    end if
  end if
end if
```

Scriptereignis HTML Verschlagwortungsanzeige

Ab der Version 4.00.080 gibt es im ELO eine neue Option zur Anzeige der Verschlagwortungsdaten parallel zum Dokument. Die Anzeige können Sie über ein HTML Dokument selber definieren. Dabei können Sie für den Maskentyp X eine Datei templ_X.htm (also z.B. templ_23.htm für Maske 23) im Postboxverzeichnis hinterlegen, welche die Verschlagwortungsanzeige enthält. Weiterhin können Sie eine Datei templ_default.htm für alle Masken hinterlegen, die keine spezielle Beschreibung benötigen. In diesen Dateien werden für die aktuellen Daten entsprechende Platzhalter (in Form eines HTML Kommentars, so dass Sie die Seiten mit einem beliebigen HTML Editor erstellen können) hinterlegt. Diese werden dann zur Anzeige gegen die richtigen Daten ersetzt und in einem Browserfenster im ELO Client angezeigt.

```
…
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_1--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_1--></td>
…
```

Die beiden oben aufgeführten Zeilen zeigen die Bezeichnung (<!—ELO_N_1→) und den Inhalt (<!—ELO_T_1→) der Indexzeile 1 in zwei Zellen einer Tabellenzeile an. Alle ELO Platzhalter beginnen mit einer HTML Kommentareinleitung <!--, gefolgt von dem festen Text ELO_. Danach folgt die Information, ob es sich um die Bezeichnung (N) oder den Inhalt (T) handelt. Zum Schluß kommt noch die Nummer der Indexzeile und der Kommentar wird abgeschlossen.

Das Kennzeichen für die Indexzeile kann die Zahlen 1 bis 50 (für die 50 Indexzeilen) umfassen, weiterhin stehen noch folgende Buchstaben zur Verfügung:

A	Ablagedatum	<!—ELO_T_A→
B	ELO interne Nummer der Dateianbindung	<!—ELO_T_B→
D	Dokumentendatum	<!—ELO_T_D→
E	Name des Eigentümers	<!—ELO_T_E→
I	ELO interne Nummer der Dokumentendatei	<!—ELO_T_I→
K	Kurzbezeichnung	<!—ELO_T_K→
M	Maskenname	<!—ELO_T_M→
O	ELO interne Nummer des logischen Eintrags	<!—ELO_T_O→
S	ELO interne Nummer der Signaturdatei	<!—ELO_T_S→
T	Dokumententyp	<!—ELO_T_T→
V	Verfallsdatum	<!—ELO_T_V→

Es gibt zudem noch die Möglichkeit, in Abhängigkeit davon, ob ein Wert eingetragen ist oder nicht, Teile des HTML Dokuments komplett weg zu lassen. Gerade bei den Indexzeilen kann es viel Platz sparen, wenn die leeren Zeilen nicht angezeigt werden. Hierzu fassen Sie den kompletten Bereich in eine Klammer aus den Kommentarkennzeichen <!—ELO_B_xxx→ und <!—ELO_E_xxx→ ein (xxx steht für die Zeilennummer oder eines der oben aufgeführten Spezialzeichen).

```
…
<!--ELO_B_1--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_1--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_1--></td>
</tr><!--ELO_E_1-->
…
```

In diesem Beispiel wird die Indexzeile 1 nur dann ein Form einer Tabellenzeile eingefügt, wenn der Text in der Indexzeile 1 nicht leer ist. Für die numerischen Felder mit den ELO internen Objektnummern gilt die 0 (kein Dokument zugeordnet) als „leer“. Tabellenzeilen, die als Unsichtbar markiert sind oder für den Anwender keinen Lesezugriff erlauben, werden ebenfalls nicht angezeigt.

Prinzipiell sind in der HTML Template-Datei alle zulässigen HTML Konstrukte (einschließlich CSS und Java Script) erlaubt. Bedenken Sie jedoch, dass die Scripting Funktionen auf einigen Browsern abgeschaltet sind. Weiterhin müssen Sie beachten, dass die Quelle aus einer Datei und nicht von einem Server kommt, alle aktiven Inhalte (Active Server Pages, Server Side Includes) würden nicht bearbeitet werden.

Das Scriptereignis wird aufgerufen bevor die Template Datei geladen wird. Zum Aufrufzeitpunkt sind die normalen Objektproperties zum anzuzeigenden Dokument wie nach einem PrepareObjectEx(Id...) geladen und das Property ViewFileName enthält den Namen der zu ladenden Template Datei (z.B. c:\temp\templ_7.htm). Die Template Datei können Sie nun im Script auf eine andere Datei umleiten (durch das Setzen des Properties ViewFileName). Zudem können Sie die Werte der Indexzeilen bei Bedarf noch verändern (SetObjAttrib...).

Beispiel für eine Anzeige der Kurzbezeichnung und der ersten 11 Indexzeilen (wenn sie nicht leer sind):

```
<html><head>
<title>ELOoffice Standardmaske</title>
</head>
<body bgcolor="#f0f0f0">

<table cellspacing=2 cellpadding=3 border=0 width=100%>
<tr><td width="80" bgcolor="#d8d8d8" valign="top"><span style="font-
size:8pt"><!--ELO_T_D--></span></td>
<td bgcolor="#d8d8d8"><h4><!--ELO_T_K--></h4></td>
</tr>

<!--ELO_B_1--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_1--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_1--></td>
</tr><!--ELO_E_1-->

<!--ELO_B_2--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_2--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_2--></td>
</tr><!--ELO_E_2-->

<!--ELO_B_3--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_3--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_3--></td>
</tr><!--ELO_E_3-->

<!--ELO_B_4--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_4--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_4--></td>
</tr><!--ELO_E_4-->

<!--ELO_B_5--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_5--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_5--></td>
</tr><!--ELO_E_5-->

<!--ELO_B_6--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_6--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_6--></td>
</tr><!--ELO_E_6-->

<!--ELO_B_7--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_7--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_7--></td>
```

```
</tr><!--ELO_E_7-->

<!--ELO_B_8--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_8--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_8--></td>
</tr><!--ELO_E_8-->

<!--ELO_B_9--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_8--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_9--></td>
</tr><!--ELO_E_9-->

<!--ELO_B_10--><tr>
<td width="80" bgcolor="#d8d8d8"><!--ELO_N_10--></td>
<td bgcolor="#d8d8d8"><!--ELO_T_10--></td>
</tr><!--ELO_E_10-->


</table>

</body>
</html>
```

Spezielle Script Ereignisse

ELO kennt eine Reihe von speziellen Scripten, die nicht konfiguriert werden müssen sondern anhand ihres Namens erkannt werden. Sobald so ein Script mit dem entsprechenden Namen im ELOScripts Verzeichnis hinterlegt wird, rufen alle ELO Clients dieses bei der entsprechenden Aktion auf.

Dokument einfrieren (ELO_Freeze)

Der ELO Client kennt eine Funktion zum einfrieren von Dokumenten. Hierbei wird der Standarddrucker auf den ELO TiffPrinter umgeschaltet, das Dokument eingelesen und per ShellExecute("print"...) vom Windows ausgedruckt. Falls der ShellExecute Befehl das nicht korrekt oder unvollständig durchführt, kann der Druck auch über ein Script durchgeführt werden. Hierzu legen Sie ein Script mit dem Namen ELO_Freeze an welches die entsprechenden Aktionen durchführt.

Vor dem Aufruf setzt der Client den Parameter ActivePostFile auf den Dateinamen des auszudruckenden Dokuments. Über den Parameter ActionKey teilt das Script mit, ob es den Druck selber übernehmen will. Wenn der Wert auf 0 steht (default-Voreinstellung), heisst das, dass ELO die Ausgabe übernehmen soll. Der Wert 1 bestätigt, dass das Script den Ausdruck veranlasst hat und der Wert 2 ist eine Fehlermitteilung, die zum Abbruch der Operation führt.

Beispiel:

Bei XLS Dateien wird von Excel per ShellExecute nur die aktuelle Tabelle ausgedruckt. Wenn das Dokument mehrere Tabellen hat, bleiben die weiteren unberücksichtigt. Das folgende Beispielscript sorgt dafür, dass alle Tabellen ausgegeben werden.

```
Set Elo=CreateObject("ELO.office")
FName=Elo.ActivePostFile

Elo.Status "Datei: " & FName

if UCase(Right(FName,4))=".XLS" then
  Set objXL = CreateObject("Excel.Application")
  call objXL.Workbooks.Open( FName )
  objXL.Visible = TRUE
  call objXL.ActiveWorkbook.PrintOut()
  objXL.ActiveWorkbook.Close
  Elo.ActionKey=1
end if
```

Thesaurus (ELO_Thesaurus)

ELO kann nicht nur einen Thesaurus sondern eine beliebige Anzahl davon verwalten. Die Auswahl, in welcher Indexzeile welcher Thesaurus verwendet werden soll, kann durch ein Script erfolgen. Dieses kann die Auswahl dann Anhand der Indexzeile (Property ScriptActionKey), Anhand des aktuellen Anwenders, der aktuellen Maske oder Anhand des Gruppennamens der Indexzeile (Property ActivePostFile) durchführen. Die Nummer des zu verwendenden Thesaurus wird über das Property ActionKey zurückgegeben.

Beispiel:

```
Set Elo=CreateObject("ELO.office")

Elo.Status "Gruppe:" & Elo.ActivePostFile & " Zeile:" & Elo.ScriptActionKey
If Elo.ObjMaskNo=3 then
  Elo.ActionKey=1
Else
  Elo.ActionKey=2
```

```
End if
```

Wiedervorlagebenachrichtigungen (ELO_OUTLOOK)

Beim Anlegen eines Wiedervorlagetermins kann ELO im Outlook eine entsprechende Aufgabe erzeugen. Das geht aber nur dann, wenn Outlook an einem Exchange Server betrieben wird und die entsprechenden Berechtigungen freigeschaltet sind (was gerade in größeren Firmen nicht der Fall ist, da ja nicht jeder Anwender für jeden beliebigen anderen Anwender ein Termin anlegen können soll). Für den Fall, dass die vorhandene Anbindung nicht eingesetzt werden kann, gibt es die Möglichkeit eine beliebige Aktion über dieses Script zu initiieren. Hierüber ist dann auch die Benachrichtigung an ein beliebiges e-mail System möglich.

Beispiel:

```
'================================================
' ELOoffice_Outlook.VBS
'================================================
'
' ELO Wiedervorlagen und Workflows in Outlook
'
'------------------------------------------------
'
' Das Skript wird vom ELOoffice Client
' aufgerufen, wenn Wiedervorlagetermine oder
' Workflows in die Outlook Aufgabenliste
' eingetragen oder Outlook-Mails generiert
' werden.
'
'------------------------------------------------
'
' © 2001 ELO Digital Office GmbH
'
' Letzte Änderung: 24.10.2001
'
'------------------------------------------------

Option Explicit

Dim oElo, oOutlook

Function GetOutlookObj
  GetOutlookObj=false
  Set oOutlook=CreateObject("Outlook.Application")
  If Err.Number=0 Then
    GetOutlookObj=true
  End If
End Function

'----------------------------
' Löschen Outlook Taskeintrag
'----------------------------
Sub WVDeleteOutlook
  Dim oMAPI, oObjRecipient, oFolder, oItems, oUserProperty, oItem
  Dim i, iID

  Set oMAPI=oOutlook.GetNameSpace("MAPI")
  Set oObjRecipient=oMAPI.CreateRecipient(oElo.DelOutlookName)
  Set oFolder=oMAPI.GetSharedDefaultFolder(oObjRecipient,13)
  Set oItems=oFolder.Items
  For i=1 To oItems.Count
    Set oItem=oItems(i)
    iID=oItem.UserProperties(1).Value
    If iID=oElo.WvIdent Then
      oItem.Delete
      Exit For
```

```
    End If
  Next

End Sub

' ---------------------------------------
' Wiedervorlagetermin als Outlook Aufgabe
' ---------------------------------------
Sub WVInOutlookTask
  Dim oMAPI, oObjRecipients, oObjRecipient, oFolder, oItems
  Dim oUserProperties, oUserProperty, oItem
  Dim i, iID

  Set oMAPI=oOutlook.GetNameSpace("MAPI")
  Set oObjRecipient=oMAPI.CreateRecipient(oElo.OutlookName)
  Set oFolder=oMAPI.GetSharedDefaultFolder(oObjRecipient,13)
  Set oItems=oFolder.Items
  If oElo.WVNew=1 Then
    Set oItem=oItems.Add
    oItem.Subject=oElo.WVShort
    oItem.Body=oElo.WVDesc
    Set oObjRecipients=oItem.Recipients
    oObjRecipients.Add(oElo.OutlookName)
    oItem.Importance=oElo.WVPrio
    oItem.StartDate=oElo.WVDate
    Set oUserProperties=oItem.UserProperties
    Set oUserProperty=oUserProperties.Add("WvId",3)
    oUserProperty.Value=oElo.WVIdent
    oItem.Save
  Else
    For i=1 to oItems.Count
      Set oItem=oFolder.Items(i)
      Set oUserProperty=oItem.UserProperties(1)
      iID=oUserProperty.Value
      If iID=oElo.WvIdent Then
        oItem.Subject=oElo.WVShort
        oItem.Body=oElo.WVDesc
        Set oObjRecipient=oItem.Recipients(1)
        oObjRecipient.Delete
        Set oObjRecipients=oItem.Recipients
        oObjRecipients.Add(oElo.OutlookName)
        oItem.Importance=oElo.WVPrio
        oItem.StartDate=oElo.WVDate
        oItem.Save
        Exit For
      End If
    Next
  End If
End Sub

' ------------------------------------
' Wiedervorlagetermin als Outlook Mail
' ------------------------------------
Sub WVInOutlookMail
  Dim oMail

  Set oMail=oOutlook.CreateItem(0)
  oMail.Recipients.Add(oElo.OutlookName)
  oMail.Subject="[" & Chr(oElo.WvPrio+64) & "] -> " & oElo.WVDate & ": " &
oElo.WvShort & " / (ELOoffice)"
  oMail.Body=oElo.WvDesc
  oMail.Send
End Sub
```

```
' -------------------------
' Workflow als Outlook Task
' -------------------------
Sub WFInOutlookTask
  Dim oMAPI, oObjRecipients, oObjRecipient, oFolder, oItems, oItem

  Set oMAPI=oOutlook.GetNameSpace("MAPI")
  Set oObjRecipient=oMAPI.CreateRecipient(oElo.WFOwner)
  Set oFolder=oMAPI.GetSharedDefaultFolder(oObjRecipient,13)
  Set oItems=oFolder.Items
  Set oItem=oItems.Add
  oItem.Subject="[W] " & oElo.WFFlowName & ": " & oElo.WFName & " /
(ELOoffice)"
  oItem.Body=oElo.WFComment
  oItem.StartDate=Date
  Set oObjRecipients=oItem.Recipients
  oObjRecipients.Add(oElo.WFOwner)
  oItem.Save
End Sub

' -------------------------
' Workflow als Outlook Mail
' -------------------------
Sub WFInOutlookMail
  Dim oMail

  Set oMail=oOutlook.CreateItem(0)
  oMail.Recipients.Add(oElo.WFOwner)
  oMail.Subject="[W] " & oElo.WFFlowName & ": " & oElo.WFName & " /
(ELOoffice)"
  oMail.Body=oElo.WFComment
  oMail.Send
End Sub

'-----------------------------------------------
'-----------------------------------------------
Sub Main

  Set oElo=CreateObject("ELO.office")

  If Not GetOutlookObj Then
    oElo.ScriptActionKey=-1
    Exit Sub
  End If

  Select Case oElo.ActionKey
    Case 1
      WVDeleteOutlook
      WVInOutlookTask
    Case 2
      WVInOutlookTask
    Case 3
      WVInOutlookMail
    Case 4
      WFInOutlookTask
    Case 5
      WFInOutlookMail
  End Select

End Sub

Main
```

Automatische Aktionen beim Programmstart (ELO_START)

Obwohl es explizite Events "Beim Betreten des Archivs" und "Beim Verlassen des Archivs gibt, haben wir zusätzlich noch ein automatisch startendes Script (ELO_START.VBS) definiert, welches bei diesen Ereignissen aufgerufen wird, ohne dass es im Client eingestellt werden muss. Das Script kann die beiden Zustände anhand des Properties ActionKey (1: Betreten, 2: Verlassen) unterscheiden. Falls sowohl das explizite Script-Event eingetragen ist wie auch das automatisch startende Script, dann werden beide Scripts ausgeführt. Beim Start zuerst das automatische, dann das explizite und beim Verlassen genau anders rum.

Sonderbehandlung bei der Neuablage von Dokumenten

Bei der Neuablage eines Dokuments in das Archiv kann ein Skript die Kontrolle über den Ablagevorgang übernehmen. Der Mechanismus wird aktiv, sobald der Anwender ein neues Dokument in das Archiv legt, sei es aus der Postbox oder per Drag & Drop aus dem Windows Explorer. Unmittelbar bevor ELO die Ablagemaske anzeigt, wird geprüft, ob ein Skript mit dem Namen „ELO_EXT_[Extension]“ im System zur Verfügung steht, eine Sonderbehandlung kann also an einen Dokumententyp gebunden werden (z.B. „ELO_EXT_DWG“ zur Behandlung von AutoCAD Zeichnungen). Ist das entsprechende Skript vorhanden, wird es gestartet. Das ELO Property *ViewFileName* enthält den Namen der abzulegenden Datei, das Property *ObjMainparent* die ID des Aktenstrukturelements, in das der Anwender das Dokument gelegt hat. Das Skript kann nun die komplette Dokumentenablage in Eigenregie durchführen, z.B. können mit dieser Methode weitere, mit dem Original-Dokument verknüpfte Dokumente in das Archiv eingebracht werden. Die Verschlagwortung muss innerhalb des Skripts erfolgen. Über das Property *ScriptActionKey* wird ELO darüber informiert, dass der Ablagevorgang vom Skripts übernommen wurde, so dass keine weiteren Aktionen von ELO selbst mehr nötig werden. Das Property wird hierzu auf den Wert -30 gesetzt.

Dialog „Dokument vom Backup“ unterdrücken (ELO_READDOC)

Wenn in der Archivansicht ein Dokument nicht geladen werden kann, erscheint ein Dialog welcher dem Anwender die Auswahl zwischen Abbruch, erneut versuchen und vom Backup neu Laden bietet. Über das Script ELO_READDOC können Sie diesen Dialog unterdrücken und die Anwenderentscheidung durch ein eigenes Script herbeiführen. Gesteuert wird dieses Verhalten über das Property ScriptActionKey. Folgende Werte stehen Ihnen zur Verfügung:

0: Normalen Dialog anzeigen
1: Kein Dialog, weiter mit „Abbruch“
2: Kein Dialog, weiter mit „Retry“
3: Kein Dialog, weiter mit „vom Backup“

Achtung: diese Abfrage läuft in einer Schleife bis das Dokument entweder geladen werden konnte oder der Anwender sich für „Abbruch“ entschieden hat. Wenn Sie im Script immer auf „Retry“ oder „Backup“ gehen und das Dokument nicht geladen werden kann, bleibt das Programm an dieser Stelle hängen.

Skripte aufrufen

Die innerhalb der Skriptverwaltung erstellten Skripte können auf mehrere Arten aufgerufen werden:

-Aufruf aus dem Kontextmenü der Symbolleiste:

Drücken Sie in einer beliebigen Ansicht von ELO die rechte Maustaste, wenn sich Ihre Maus auf der Buttonleiste befindet, im daraufhin erscheinenden Menü können Sie eines Ihrer Skripte starten.

-Aufruf über User-Button:

In jeder Ansicht stehen 8 belegbare Buttons zur Verfügung, auf die jeweils ein Skript gelegt werden kann. Hierzu muß zunächst das Skriptmenü (rechte Maustaste in der Buttonleiste) aufgerufen werden, wenn sich die Maus über einem der 4 Buttons befindet. Wird nun ein Menüpunkt (=Skriptname) bei gleichzeitig gedrückter Strg-Taste angewählt, wird dieses Skript auf den Button der aktuellen Ansicht gelegt. Die aktuelle Belegung wird sichtbar, wenn sich die Maus über dem Button befindet. Soll ein Skript vom Button entfernt werden, klicken Sie den Button bei bei gleichzeitig gedrückter Strg-Taste an.
Nach einer Neuinstallation von ELO sind die Buttons standardmäßig nicht sichtbar und müssen gegebenenfalls mit Hilfe der Funktion *Ansicht-Werkzeugleiste konfigurieren* eingeschaltet werden.
Zur Darstellung eines Icons auf dem Skriptbutton muss eine BMP-Datei mit einer Größe von 24 x 24 Pixel angelegt und unter dem Namen des Skriptes im Verzeichnis der ELO Skripte abgelegt werden.

-Aufruf über Kontextmenü:

In jeder Ansicht kann das jeweilige Kontextmenü um Skriptaufrufe ergänzt werden. Soll ein Skript in das Kontextmenü einer bestimmten Ansicht eingebaut werden, muss diese Ansicht zunächst angewählt werden. Im Kontextmenü der Symbolleiste wird dann das entsprechende Skript selektiert, während gleichzeitig die Tasten *Shift* und *Strg* gedrückt werden. Zum Entfernen eines Skripts aus dem Kontextmenü wird das Skript im Kontextmenü mit gedrückter *Strg*-Taste angeklickt.

-Aufruf über ELO-Menüpunkte oder -Buttons:

Jeder Menüeintrag und jeder Button in der Hauptansicht von ELO kann mit einem eigenen Skript belegt werden, das dann anstelle der Originalfunktion aufgerufen wird. Innerhalb des Skripts kann gesteuert werden, ob die Original ELO-Funktion nach dem Ende des Skriptes noch aufgerufen wird oder ob dies unterbleiben soll.
Die Skript-Zuweisung geschieht über den internen Namen eines Menüeintrags oder Buttons. Der Name des Skripts muss mit einem "." beginnen, gefolgt von dem Namen des Controls (Menüeintrag oder Button). Eine Liste der Controls ist im Anhang der Dokumentation der ELO Automationschnittstelle zu finden.
Beim ersten Aufruf des Skripts besitzt das Property *ActionKey* den Wert 40. Das Skript kann über das Setzen des Properties *ScriptActionKey* den weiteren Ablauf steuern:
20 = Original ELO-Funktion wird nach dem Ende des Skriptes aufgerufen
21 = Original ELO-Funktion wird nach dem Ende des Skriptes aufgerufen, danach wird erneut das Skript gestartet, diesmal mit dem *ActionKey* 0

Formularerkennungs-API

Übersicht

Die Formularerkennung unter ELO verläuft in zwei bis vier Schritten. In einem ersten Schritt werden die Recheckbereiche des Dokuments markiert, welche zur Klassifikation des Dokumententyps notwendig sind. Typische Bereiche sind hier die Texte „Rechnung“ oder „Lieferschein“ oder aber auch ein Firmenname (keine grafischen Elemente). Diese Bereiche werden dann durch die OCR Software bearbeitet, erkannt und in einer Textliste abgelegt. Hierzu gibt es das erweiterte OCR API.

In einem zweiten Schritt wird jetzt für jeden Dokumententyp nach kennzeichnenden Mustern in den erkannten Texten gesucht. Wenn eine bestimmte Rechnung an einer vorgegebenen Stelle (in der oben aktivierten Rechteckliste) den Text „Rechnung“ folgend von einer Rechnungsnummer hat, muss zur Erkennung dieses Formulars also nur der entsprechende Rechnungstext auf dieses Muster hin kontrolliert werden. Hierzu wird das Mustererkennungs-API eingesetzt.

Wenn der Formulartyp erkannt wurde, kann es sich als notwendig erweisen zusätzliche Textbereiche einzulesen. Es können also zu dem Rechnungsformular noch die Bestellnummer, Kundennummer oder der Rechnungsbetrag erkannt werden. Dieser Schritt kann im Prinzip direkt mit dem Schritt 1 stattfinden. Beachten Sie aber dabei, dass Sie in diesem Schritt auf Verdacht alle Rechteckbereiche für alle möglichen Dokumenttypen untersuchen müssten. Da das sehr (zeit)aufwendig werden kann, ist es bei einer größeren Anzahl von Typen im Allgemeinen sinnvoller, dieses auf einen eigenen, dritten Schritt zu verlagern.

Nachdem alle Textbereiche eingelesen worden sind und der Dokumententyp feststeht, muss nun noch gezielt die Indizierung aus den Textblöcken extrahiert werden. Bei diesem vierten Schritt (der bei einfachen Anwendungen möglicherweise schon durch Schritt Zwei abgedeckt wurde) kommt wieder das Mustererkennungs-API zum Einsatz.

Befehle des Mustererkennungs-API

Funktion OcrAddRect
Funktion OcrAnalyze
Funktion OcrClearRect
Funktion OcrGetPattern
Funktion OcrGetText
Funktion OcrPattern

Die Verwendung dieser Befehle wird unter den folgenden Beispielen vorgeführt. Die genauen Parameter entnehmen Sie bitte der Befehlsliste.

Beispiele

Beispiel 1: Rechnungs-Erkennung

Dieses erste Beispiel geht von einem ganz einfachen Fall aus. Es gibt nur zwei mögliche Dokumententypen, eine Rechnung und den Rest der Welt. Indiziert werden soll nur die Rechnungsnummer.

Im ersten Schritt wird nun über das OCR-API die Rechteckliste mit dem Bereich für den Rechnungstext mit der dazugehörenden Nummer eingestellt und die Erkennung gestartet.

```
x=ELO.OcrClearRect()
x=ELO.OcrAddRect("500,10,999,100")
x=ELO.OcrAnalyze(FileName,0)
```

Im zweiten Schritt wird nun kontrolliert, ob es sich bei dem Formular um eine Rechnung handelt. Hierzu wird kontrolliert, ob im Textblock 1 der Text „Rechnung“, gefolgt durch die Rechnungsnummer, erkannt werden kann.

```
CntRechnung=ELO.OcrPattern(10,"*'Rechnung'_N*", ELO.OcrGetText(0))
```

Gesucht wird hier nach einem Muster aus fünf Teilen:

1. Ein beliebiger, möglicherweise auch leerer, Vorspann aus beliebigen Zeichen (z.B. weil in das Rechteck durch unsauberen Scannereinzug von oben fremde Zeichen hereinragen).
2. Der Text ’Rechnung’
3. Eine beliebig lange, möglicherweise auch leere, Folge von Leerzeichen
4. Eine Nummer (die Rechnungsnummer)
5. Beliebiger weiterer Text (z.B. auch die durch Zeichen die nicht zum eigentlichen Textbereich gehören aber in das Erkennungsrechteck hineinragen).

Wenn das Muster erkannt worden ist, gibt die Funktion den Wert 5 (=Anzahl der Musterteile) zurück, im Fehlerfalle erhalten Sie einen negativen Wert. Der Text zu den einzelnen Musterteilen steht danach in einem Textfeld zur Verfügung.

```
If CntRechnung=5 then
   ’ es ist eine Rechnung, 5 Musterblöcke wurden erkannt
    ELO.PrepareObjectEx(0,254,RechnungsMaskNo)
    ELO.SetObjAttrib(0, ELO.OcrGetPattern(3))
      …
end if
```

Die Vier-Schritte Arbeit reduziert sich in diesem einfachen Beispiel auf zwei Schritte. Schritt 3 entfällt, da keine weiteren OCR Bereiche eingelesen werden müssen und Schritt 4 entfällt, weil die Indizierung bereits während der Klassifikation erkannt wurde. Durch die Kontrolle, ob es sich um eine Rechnung handelt, wurde gleich auch die Rechnungsnummer in das Mustertextfeld eingelesen und kann direkt verwendet werden.

Beispiel 2: Rechnung/ Lieferschein Erkennung

Dieses Beispiel geht immer noch von einem recht einfachen Fall aus. Es gibt nur zwei mögliche Dokumententypen, eine Rechnung und ein Lieferschein. Indiziert werden soll jeweils nur die Rechnungs- oder Lieferscheinnummer.

Im ersten Schritt wird nun über das OCR-API die Rechteckliste mit den beiden Bereichen für den Rechnungstext bzw. den Lieferscheintext, jeweils mit der dazugehörenden Nummer, eingestellt und die Erkennung gestartet.

```
x=Elo.OcrClearRect()
x=Elo.OcrAddRect("500,250,999,390")
x=Elo.OcrAddRect("500,10,999,100")
x=Elo.OcrAnalyze(FileName,0)
```

Als nächstes wird kontrolliert, ob es sich bei dem Formular um eine Rechnung oder um einen Lieferschein handelt. Hierzu wird geprüft, ob im Textblock 1 der Text „Rechnung“ erkannt werden kann oder ob im Textblock 2 der Text „Lieferschein“ vorhanden ist.

```
CntRechnung=Elo.OcrPattern(10,"*’Rechnung’*", Elo.OcrGetText(0))
CntLieferschein=Elo.OcrPattern(10,"*’Lieferschein’*", Elo.OcrGetText(1))

If CntRechnung<0 then
   ’ es ist keine Rechnung
  If CntLieferschein<0 then
    ’ es ist auch kein Lieferschein, dann wird auch nichts gemacht
    DocType=0
  else
```

```
    ' Lieferschein
    DocType=1
  End if
Else
  ' es ist eine Rechnung
  if CntLieferschein<0 then
    ' es ist wirklich nur eine Rechnung
    DocType=2
  Else
    ' es ist eine Rechnung und ein Lieferschein - hier liegt ein Fehler
vor
    DocType=0
  End if
End if
```

Es folgt nun das Eintragen der Indizierung. Weitere Rechtecke werden in diesem einfachen Beispiel nicht eingelesen, die Rechnungs- bzw. Lieferscheinnummer ist bereits per OCR erkannt worden.

```
Select Case DocType
  case 1 `Lieferschein
    ELO.PrepareObjectEx(0,254,LieferscheinMaskNo)
    CntLieferschein=Analyze( "*'Lieferschein'_N*", OcrText(1) )
    If CntLieferschein=5 then
      ELO.SetObjAttrib(0, Elo.OcrGetPattern (3))
      ...
    end if
  case 2 `Rechnung
    ELO.PrepareObjectEx(0,254,RechnungsMaskNo)
    CntRechnung=Analyze(  "*'Rechnung'_N*", OcrText(0) )
    If CntRechnung=5 then
      ELO.SetObjAttrib(0, Elo.OcrGetPattern (3))
      …
    end if
end select
```

Beispiel 3: Formularerkennung Messe-Demo

Das folgende Beispiel ist ein komplettes Script zur Erkennung von drei unterschiedlichen Formularen und automatischer Ablage im Archiv (bei Bedarf wird der Ordner gleich mit angelegt).

```
'OCRELO.VBS  24.08.2000
'-------------------------------------------------------------------
' © 2000 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo.info)
'-------------------------------------------------------------------
' Dieses Skript untersucht die Postboxdokumente auf bestimmte
' Texte welche Rechnungsnummern kennzeichnen und legt die
' erkannten Rechnungen dann in den dazu passenden Registern
' ab (die bei Bedarf automatisch erzeugt werden)
'
' -------------------------------------------------------------------

set Elo=CreateObject("ELO.office")
MaskNo=Elo.LookupMaskName("ELORechnung")

' laufe über alle Postboxeinträge (maximal 200)
Elo.SelectView(3)

' erstmal alle Dokumente analysieren
```

```
Elo.Status "Dokumente analysieren"
Elo.UnselectAllPostboxLines
for i=0 to 300
  res=Elo.PrepareObject( -1,i,MaskNo )
  if res=-6 then
    exit for
  end if
  if Elo.ObjShort="" then
    fname=Elo.ActivePostFile
    if UCase(Right(fname,4))=".TIF" then
      x=Elo.UpdatePostboxEx( 20,i )
      x=Elo.Status(fname)
      Analyze i, fname
      Elo.UnselectPostboxLine(i)
    end if
  end if
next

' und nun die erkannten Dokumente ins Archiv übertragen
for j=i to 0 step -1
  Move(j)
next

' kurz noch die Postboxansicht saubermachen ...
x=Elo.UpdatePostboxEx(0,0)
Elo.Status("Fertig")
' ... und fertig


' Hilfsfunktionen
'
' diese Funktion überträgt ein erkanntes Dokument ins Archiv
sub Move( iPostLine )
  res=Elo.PrepareObject( -1, iPostLine, 0 )
  if res>0 then
    Text=Elo.ObjShort
    Elo.Status("Ablegen: " & Text)
    Datum=Mid(Text,12,10)
    Kdnr=Mid(Text,23,10)
    Renr=Trim(Left(Text,10))
    if Len(Datum)=10 and Len(KdNr)>0 then
      if Left(Renr,3)="300" then
        Renr=Renr & " (Gutschrift)"
      else
        Renr=Renr & " (Rechnung)"
      end if
      Elo.ObjShort=Renr
      iRet=Elo.AddPostBoxFile("")
      RegId=CheckRegister( Datum, Kdnr )
      if RegId>0 then
        x=Elo.MoveToArchive( "#" & RegId )
      end if
    end if
  end if
end sub

' diese Funktion prüft, ob das Zielregister für ein Dokument
' vorhanden ist und legt es bei Bedarf an
function CheckRegister( Datum, Kundennr )
  RegId=Elo.LookupIndex( "RELO=" & Right(Datum,4) & ":" & Kundennr )
```

```
  if RegId<1 then
    ' Register noch nicht vorhanden, wird nun angelegt
    OrdnerId=Elo.LookupIndex( "¶ELO Rechnungen¶" & Right(Datum,4) )
    if OrdnerId>0 then
      ' Ordner gefunden, nun das Register erzeugen
      if Elo.PrepareObjectEx( 0,253,0 ) then
        Elo.ObjShort=Kundennr
        x=Elo.SetObjAttrib(0,Right(Datum,4) & ":" & Kundennr)
        x=Elo.SetObjAttribKey(0,"RELO")
        Elo.ObjFlags=4
        Elo.ObjIndex="#" & OrdnerId
        Elo.UpdateObject()
        RegId=Elo.GetEntryId(-2)
      end if
    end if
  end if
  CheckRegister=RegId
end function

' diese Funktion führt eine OCR Analyse über das aktuelle Postboxdokument
durch
' und legt die Rechnungsnummer in der Kurzbezeichnung ab wenn ein
bekannter
' Dokumententyp vorgefunden wurde
sub Analyze( iPostLine, FileName )
  x=Elo.OcrClearRect()
  x=Elo.OcrAddRect("500,250,999,390")
  x=Elo.OcrAddRect("500,10,999,100")
  x=Elo.OcrAnalyze(FileName,0)
  if x<0 then
    exit sub
  end if

  'x=Elo.OcrPattern( 10,"*", Elo.OcrGetText(0) )
  'MsgBox Elo.OcrGetPattern(0)
  'x=Elo.OcrPattern( 10,"*", Elo.OcrGetText(1) )
  'MsgBox Elo.OcrGetPattern(0)

  Elo.ObjShort=""
  Elo.ObjMemo=""
  found=false
  if not found then
    x=Elo.OcrPattern(
10,"'Rechnung'L'Nummer:'NL'Datum:'*L'Auftragsnr.:'NL'Kunden-Nr.:'NL*",
Elo.OcrGetText(0))
    if x>0 then 'ELO Rechnung
      Elo.ObjShort=Left(Elo.OcrGetPattern(3)&"          ",10) & " "
&Elo.OcrGetPattern(6) & " " & Elo.OcrGetPattern(12)
      x=Elo.SetObjAttrib(0,Elo.OcrGetPattern(12))
      x=Elo.SetObjAttrib(1,Elo.OcrGetPattern(3))
      x=Elo.SetObjAttrib(2,Elo.OcrGetPattern(9))
      Elo.ObjXDate=Elo.OcrGetPattern(6)
      found=true
    end if
  end if
  if not found then 'ELO Gutschrift
    x=Elo.OcrPattern( 10,"*'Gutschrift'L'Nummer:'NL'Datum:'*L'Auftrags-
Nr.:'NL'Kunden-Nr.:'N*", Elo.OcrGetText(0))
    if x>0 then
```

```
      Elo.ObjShort=Left(Elo.OcrGetPattern(4)&"          ",10) & " "
&Elo.OcrGetPattern(7) & " " & Elo.OcrGetPattern(13)
      x=Elo.SetObjAttrib(0,Elo.OcrGetPattern(13))
      x=Elo.SetObjAttrib(1,Elo.OcrGetPattern(4))
      x=Elo.SetObjAttrib(2,Elo.OcrGetPattern(10))
      Elo.ObjXDate=Elo.OcrGetPattern(7)
      found=true
    end if
  end if
  if not found then  'NOKIA Rechnung
    x=Elo.OcrPattern( 10,"*'Rechnung'n*'Kundennummer'N*",
Elo.OcrGetText(1) )
    if x>0 then
      Elo.ObjShort=Left(Elo.OcrGetPattern(2)&"          ",10) & "
01.01.2000 " & Elo.OcrGetPattern(5)
      x=Elo.SetObjAttrib(0,Elo.OcrGetPattern(5))
      x=Elo.SetObjAttrib(1,Elo.OcrGetPattern(2))
      Elo.ObjXDate="01.01.2000"
      found=true
    end if
  end if

  if found then
    iRet=Elo.AddPostBoxFile("")
  end if
  Elo.Status "Erkennen  Zeile " & i & " : " & Elo.ObjShort
end sub
```

Syntax des Formatstrings der Mustererkennung

Zur Mustererkennung muss nur ein Formatstring vorgegeben werden. Die Kontrolle, ob der Text diesem Muster genügt und die Aufteilung des Textes auf die Musteranteile wird von ELO in einem Schritt durchgeführt. Beachten Sie bitte, dass ein Muster aus maximal 32 Teilen (in der Version 104, ändert sich später möglicherweise) bestehen darf. Folgende Teilmuster stehen zur Verfügung:

*	Beliebiger Text	Dieses Teilmuster akzeptiert einen beliebigen Text, er kann auch leer sein. Sie können zusätzlich eine Längenangabe vor den Stern setzen, dann muss der erkannte Text mindestens so lang sein, wie die Vorgabe fordert.
_	Leerzeichen	Dieses Teilmuster akzeptiert eine beliebige, auch leere, Folge von Leerzeichen. Eine Längenangabe vor dem Unterstrich führt dazu, dass ein Folge mit der exakten Vorgabelänge erkannt wird.
L	Zeilenwechsel	Dieses Teilmuster erkennt einen Zeilenwechsel (genau einen). Eine Zahl vor dem L (z.B.: 3L) fordert genau diese Anzahl von Zeilenwechseln.
N	Nummer	Es wird eine beliebig lange Folge von Ziffern erkannt (aber keine leere Folge). Beendet wird diese Folge durch das erste Zeichen welches keine Ziffer ist (auch Zeilenwechsel oder Leerzeichen). Falls ein Multiplikator vorangestellt wird, wird eine Ziffernfolge genau dieser Länge erkannt. Beispiel: ABC12345XYZ *N* erkennt [ABC][12345][XYZ] *3N* erkennt [ABC][123][45XYZ] *6N* erkennt nichts, da es keine Ziffernfolge dieser Länge gibt.
n	Nummer (spezial OCR)	Wie bei N (Nummer) – der Unterschied liegt darin, dass dieses Format auch ein paar Buchstaben akzeptiert, welche bestimmten Ziffern sehr ähnlich sind (O, o und Q werden als 0 (Null) erkannt, I und l werden als 1 (Eins) erkannt.
"..."	Text	Es wird der Text in den Anführungszeichen akzeptiert. Dabei muss dieser Text wirklich exakt in dieser Form vorliegen, es werden keine zusätzlichen Leerzeichen oder Zeilenwechsel erkannt. Beispiel: xyzRechnung 123 *"Rechnung"_N* erkennt [xyz][Rechnung][ ][123][] *"Rechnung"N* erkennt nichts, das Leerzeichen wurde im Formatstring nicht berücksichtigt *"RECHNUNG"_N* erkennt nicht, da Rechnung anders geschrieben ist.
'...'	Text	Wie beim doppelten Anführungszeichen, nur dass der Text nicht case sensitive erkannt wird. Im Beispiel oben würde nun also der dritte Fall erkannt werden.

Anmerkungen

Beachten Sie, dass der Erkennungsalgorithmus so lange sucht, bis es ein „match" erreicht hat oder bis es sicher ist, dass es keinen gibt. Er „fährt" sich nicht an Teillösungen fest. Eine naive Implementierung könnte bei dem Muster „*'Rechnung'_N*" und dem Text „xxx Rechnungskopie yyy Rechnung 12345 zzz" den Text „Rechnung" aus Rechnungskopie anpeilen und nachdem keine Nummer folgt aufgeben. ELO hingegen sucht nach diesem Fehlversuch weiter und erkennt den Text so wie man es erwartet.

Speziell das Muster * verursacht hohen internen Aufwand, da im Zweifelsfall viele Möglichkeiten untersucht werden müssen. Trotzdem ist es oft notwendig diesen Operator einzusetzen. Insbesondere sollten die Muster im Allgemeinen durch ein *....* eingerahmt werden. Hierdurch werden „Schmutzzeichen“ am Anfang oder Ende des Textes abgefangen. Diese werden oft durch Fremdzeilen, welche in das Erkennungsrechteck hineinragen, hervorgerufen. Trotzdem sollte man es nur dort einsetzen, wo es gerechtfertigt ist. Die überflüssige, aber scheinbar harmlose Kombination ** verändert zwar nicht das Erkennungsergebnis, hat aber äußerst nachteiligen Einfluss auf die Performance. Das Muster
** zwingt ELO bei dem Text „abcd“ zur Kontrolle der Möglichkeiten [][abcd], [a][bcd], [ab][cd], [abc][d], [abcd][].

Packen Sie nicht zuviel in ein Muster. Wenn Sie in einer Rechnung ein Rechteck mit aufeinanderfolgenden Zeilen mit Rechnungsnummer, Kundennummer, Bestellnummer und Auftragsnummer haben, dann können Sie jede Nummer einzeln Suchen. Sie können aber auch ein komplexes Suchmuster
’Rechnung’_N’Kunde’_N*’Bestellnr.’_N*’Auftrag’_N* formulieren. Im zweiten Fall würden alle Nummern in einem Durchgang erkannt werden. Sobald aber nur eine dieser Nummer von der OCR Software nicht korrekt umgesetzt wurde (z.B. Besteilnummer statt Bestellnummer), wird keine einzige Nummer mehr ermittelt werden. Falls die Indizierung für Sie nur von Interesse ist, wenn alles erkannt wurde, ist die zweite Variante angemessen. Falls Sie die Ansicht vertreten, dass möglichst viel erkannt werden sollte und nur das fehlende manuell nachgetragen werden muss, dann sollten Sie die Nummern einzeln erkennen lassen.

Gerade bei komplexen Mustern kann es leicht passieren, dass es nicht erkannt wird obwohl der Text es eigentlich zulassen sollte. Da es keinen speziellen „Musterdebugger“ gibt, hilft hier nur ein schrittweises try and error weiter. Angefangen mit dem „defekten“ Muster

’Rechnung’_N’Kunde’_N*’Bestellnr.’N*’Auftrag’_N*

können Sie in einem ersten Schritt

’Rechnung’ testen. Danach prüfen Sie
’Rechnung’_N
’Rechnung’_N’Kunde’*
’Rechnung’_N’Kunde’_N*
usw.. Das Muster wird immer um einen (oder auch mehrere) Schritt(e) verlängert. Achten Sie darauf, dass Sie das Muster immer mit einem * abschließen. Dieser * ist der match für den ganzen Rest. Wenn er fehlt, wird auch nichts erkannt werden.

Liste der verfügbaren OLE Funktionen

Allgemeine Hinweise zu den Funktionen

Die Rückgabewerte der Funktionen enthalten im Allgemeinen einen Fehlercode. In diesen Fällen signalisieren negative Werte ein Fehlschlagen der Funktion, positive Werte ein korrektes Ausführen.

ELO arbeitet intern mit sogenannten ObjektIds. Das sind (innerhalb eines Archivs) eindeutige Werte, welche jedem Eintrag beim erzeugen zugeordnet werden und über welchen die Einträger jederzeit wieder angesprochen werden können. Falls Sie irgendwelche Referenzen von ELO in Ihrem Programm benötigen und speichern wollen, sollten Sie diese ObjektId und nicht die Bezeichnung verwenden. Die ObjektId bleibt garantiert über die gesamte Lebensdauer des Eintrags unverändert erhalten, die Bezeichnung hingegen kann jederzeit vom Anwender oder von anderen Programmen verändert werden.

Property ActionKey (int, nur lesen)

Einige ELO Ereignisse lösen gemeinsam eine Scriptbehandlung aus. In diesem Fall kann über den ActionKey unterschieden werden, was für ein Ereignis eingetreten ist.
a

Dialog: Dokument bearbeiten	Dialog in Postboxansicht aufgerufen, Dokumententyp noch nicht definiert	19
	Maske betreten	20
	Maske mit Speichern verlassen	21
Event:	Maske mit Abbruch verlassen	22
	Dokumententyp geändert	23
	Buttonliste abfragen	24
Nach bearbeiten Maskenfeld	Feld Kurzbezeichnung betreten	10
	Feld Kurzbezeichnung verlassen	11
	Feld Memo betreten	12
	Feld Memo verlassen	13
	Feld Datum betreten	14
	Feld Datum verlassen	15
	Index-Eingabezeile n betreten	1000+n (n=0..49)
	Index-Eingabezeile n verlassen	2000+n
	Anwenderdefinierten Button betätigt	3000+n
Vor dem Importieren/Exportieren	Vor dem Importieren eines Aktenstrukturelements	1
	Vor dem Exportieren eines Aktenstrukturelements	2
Beim Aus-/Einchecken	Nach dem Auschecken	30
	Vor dem Einchecken, nach der Versionsabfrage	31
	Nach dem Einchecken	32
	Vor dem Auschecken	33
	Vor dem Einchecken, vor der Versionsabfrage	34
	Vor dem Verwerfen	38
	Vor dem Aktivieren/Anzeigen	80
	Vor dem Ausdrucken	81
	Register CheckIn/Out	1001 … 1005
Anwender lesen/speichern	Unmittelbar vor dem Abspeichern	20000
	Nach dem Abspeichern	20001
	Nach dem Einlesen	20010
Stichwortliste	Vor dem Bearbeiten der Stichwortliste	1
	Vor der Anzeige der Stichwortliste	2
Wiedervorlage	Termin als Aufgabe für anderen Anwender	1
	Termin als Aufgabe für sich selber	2
	Termin als e-mail	3
ClickOn Event	Auswahl eines Buttons oder Menüeintrags	40
Workflow Event	Workflowtermin nur im ELO anzeigen	3
	Workflowtermin als Aufgabe	4
	Workflowtermin als e-mail	5
Dokument einfrieren	Vor dem Ausdrucken	0
Suchen („Vor dem Sammeln der Rechercheliste“)	Vor der Suche	1
	F7-Gruppensuche	2
	Kombinierte Gruppensuche	3
	Nach der Suche	4

Diesen Zahlen ist der Wert 0x8000 überlagert, falls es sich um eine Recherchemaske und nicht um eine Datenmaske handelt. Diese Angabe ist unbedingt auszuwerten, sonst können sich unerwünschte Effekte bei der Suche von Dokumenten ergeben!

Beispiel (als Skript AutoKurzbez hinterlegen und unter Skript Events "Beim Bearbeiten der Verschlagwortung" eintragen):

```
' hier ist die Nummer der Dokumentenmaske zu hinterlegen
' die automatisch durch das Skript aus den Indexzeilen
' eins und zwei die Kurzbezeichnung einfüllen soll.
DokumentenMaske = 2

Set Elo=CreateObject("ELO.office")

if Elo.ActionKey=21 and Elo.ObjMaskNo=DokumentenMaske and Elo.ObjShort=""
then
  ' geschrieben wird beim Beenden des Verschlagwortungsdialogs, wenn die
  ' richtige Maske eingestellt ist und noch keine Kurzbezeichnung vorliegt.
  Elo.ObjShort=Elo.GetObjAttrib(0) & " " & Elo.GetObjAttrib(1)
end if
```

siehe auch:

Property ActivePostFile (AnsiString)

Das Property gibt den Zugriffspfad und Name der aktiven Postboxdatei. Gesetzt wird dieser Eintrag durch Aktionen, welche einen neuen Postboxeintrag vornehmen (z.B. Scannen oder AddPostboxFile).

Ab der Version 3.00.508 kann dieses Property auch beschrieben werden. Sie können hierüber eine Datei aktiv setzen, die bereits in der Postbox vorhanden ist.

siehe auch: AddPostboxFile
PrepareObject

Property ActiveUserId (int, nur lesen)

Das Property ActiveUserId liefert die interne ELO Anwender-Nummer des aktiven Logins.

siehe auch: LoadUserName
FindUser
SelectUser

Funktion AddAutoDlgControl (int, int, AnsiString, AnsiString)

Verfügbar ab:	**Ver 3.00.228**

Erstellt Elemente in dem vorher mit CreateAutoDlg erstelltem Dialog. **Es muss vorher ein CreateAutoDlg aufgerufen werden!**

int AddAutoDlgControl (int Type, int RasterInc, AnsiString Caption, AnsiString Default)

Type	**Erstelltes Objekt**	**Caption**	**Default**
1	Label	Text des Labels	Keine Funktion
2	CheckBox	Text hinter dem Kontrollkästchen	1 – Checked ansonst Unchecked
3	Radio Button	Text hinter dem Auswahlknopf	1 – Checked ansonst Unchecked
4	Edit Feld	Text vor dem Eingabefeld	Text des Editfeldes

RasterInc: Horizontale Position des Objekts (erlaubte Werte : 0 bis ?)

Rückgabewert :
1 – Objekt wurde erstellt.

Bsp.:

Erstellt einen Dialog mit einem Label an Oberster Position darunter ein Edit Feld mit ausgefülltem Inhalt und zeigt es an.

```
Elo.CreateAutoDlg (“Neuer Dialog“)
Elo.AddAutoDlgControl (1,0,“Oberes Label“,““)
Elo.AddAutoDlgControl (4,1,”Ihr Name”,”Musternann”)
Elo.ShowAutoDlg
```

siehe auch: CreateAutoDlg
ShowAutoDlg
GetAutoDlgValue

Funktion AddLink

Verbindet zwei ELO Objekte (Dokumenten oder Ablagestrukturelemente) durch einen logischen Link. Es können beliebige Einträge ohne Berücksichtigung der Hirarchie verlinkt werden. Zu einem Objekt können auch mehrere andere Objekte angebunden werden.

int AddLink(int StartObjekt, int Zielobjekt)

Parameter:

StartObjekt:	ObjektId des Eintrags von dem der Link ausgeht
ZielObjekt:	ObjektId des Eintrags auf das der Link zeigt

Rückgabewerte:

-2: Fehler beim Schreiben des Links
-1: Kein Arbeitsbereich aktiv
1: Link eingefügt

Verfügbar ab 3.00.270

Funktion AddNote

Funktion AddNoteEx

Fügt zu einem Dokument eine weitere Haftnotiz hinzu. Beachten Sie bitte, dass es eine maximale Anzahl von Haftnotizen zu einem Dokument gibt, weitere werden nicht angezeigt, beim Speichern aber auch nicht als fehlerhaft markiert.

Beachten Sie bitte, dass diese Funktion nicht zu einem Update der Ansicht des aktuell angezeigten Dokuments führt.

int AddNote(int ObjId, int NoteType, AnsiString NoteText)
int AddNoteEx(int ObjId, int NoteType, AnsiString NoteText, int PageNo, int x, int y)

Parameter:

ObjId:	Nummer des logischen Dokuments
NoteType:	Art der Haftnotiz
NoteText:	Textkörper der HN
PageNo:	Seitennummer 1..n für Tiff-Dokumente mit eingeblendeter Haftnotiz, 0: nur Icon
X:	x-Position im Tiff-Dokument
Y:	y-Position im Tiff-Dokument

```
Liste der NoteType
#define NOTE_TYPE_NONE           0
#define NOTE_TYPE_NORMAL         1
#define NOTE_TYPE_PERSONAL       2
#define NOTE_TYPE_STAMP          3
#define ANNOTATION_TYPE_NORMAL   4
#define ANNOTATION_TYPE_PERSONAL 5
#define ANNOTATION_TYPE_STAMP    6

#define ANNOTATION_WANG_MARKER  10
#define ANNOTATION_WANG_NOTE    11
```

Rückgabewerte:

-5: Ungültiger Notiztyp 1..3
-4: Zielobjekt ist kein Dokument
-3: Zielobjekt konnte nicht ermittelt werden
-2: Fehler beim Schreiben der Haftnotiz
-1: Kein Arbeitsbereich aktiv
>0: Speichern ok, Rückgabewert ist die interne Notiznummer

Verfügbar ab:

AddNoteEx ab 4.00.138

Funktion AddPostboxFile

Diese Funktion überträgt eine Datei in die Postbox. Hierzu muß zuerst mit PrepareObject ein neuer Eintrag vorbereitet werden, über die verschiedenen Properties die Kurzbezeichnung u.ä. Werte gesetzt werden und dann die Funktion aufgerufen werden. Es wird dann die Datei **kopiert** und in der Postbox zusammen mit der Verschlagwortung hinterlegt.

Nach Aufruf dieser Funktion wird der neue Eintrag automatisch der „aktive Postboxeintrag“, er ist dann die Quelle für einige weiterführende Funktionen.

Es gibt noch einen Sonderfall für diese Funktion: wenn der „aktive Postboxeintrag“ verändert wird und neu in der Postbox gespeichert werden soll, wird diese Funktion mit einem Leerstring als Parameter aufgerufen. Diese Möglichkeit wird z.B. von der Barcode Nachbearbeitung verwendet.

int AddPostboxFile(AnsiString SourceFile)

Parameter:

SourceFile zu kopierende Datei

Rückgabewerte:

-3: Fehler beim Speichern.
-2: Kein aktiver Postboxeintrag vorhanden
-1: Fehler beim Übertragen in die Postbox
1: ok

Beispiel: erstes Postboxdokument aufgreifen, mit der Maskennummer 2 belegen und die Kurzbezeichnung sowie das erste Indexfeld automatisch füllen. Anschließend wird das Dokument auf dem vorgegeben Pfad ins Archiv übertragen:

```
' Zielregister für das Dokument
RegisterId = "¶Schrank¶Ordner¶Register"
MaskNo = 2
Set Elo=CreateObject("ELO.office")

' Zur Sicherheit erst die Postbox aktualisieren
Elo.UpdatePostbox

x=Elo.PrepareObjectEx( -1, 0, MaskNo )
if x>0 or x=-5 or x=-7 then
  Elo.ObjShort="Test" & Time
  call Elo.SetObjAttrib(0,"Index 1")
  call Elo.AddPostboxFile("")
  x=Elo.MoveToArchive( RegisterId )
else
  MsgBox "Kein Dokument in der Postbox vorgefunden"
end if
```

siehe auch: MoveToArchive
LookupIndex
UpdateDocument
InsertAttachment

Funktion AddSignature

Diese Funktion bindet eine externe Signaturdatei an ein vorhandenes Dokument an. Es liegt dabei in der Verantwortung des Scriptes sicher zu stellen, dass die Signatur tatsächlich zu dem Dokument gehört. Im Fehlerfall wird dem Anwender sonst eine ungültige Signatur angezeigt.

int AddSignature(int ObjectId, AnsiString SignatureFile)

Parameter:

ObjectId	Logische ELO Dokumenten-Id
SignatureFile	Datei mit der Signatur-Information

Rückgabewerte:

1: ok
-1: kein Workspace offen
-2: Fehler beim Speichern der Signaturdatei

Beispiel

```
...
Result = Elo.MoveToArchive( DestPath )
If Result > 0 then
  Id = Elo.GetEntryId(-2)
  MsgBox Elo.AddSignature( Id, "d:\temp\00016882.ESG" )
End if
...
```

siehe auch:

Funktion AddSw

Diese Funktion trägt ein Stichwort in eine Stichwortliste ein. Jeder Eintrag einer Stichwortliste wird innerhalb seiner Ebene eindeutig durch eine 2-stellige Buchstabenkombination AA, AB, AC .. ZZ gekennzeichnet. Beim Eintragen eines neuen Stichwortes wird der nächste freie Eintrag ermittelt und als Ergebnis des Aufrufs zurückgegeben. Diesen Wert benötigen Sie z.B. dann, wenn Sie baumartige Untereinträge anlegen wollen.

Die Gruppe kennzeichnet die Listenzuordnung zu einer Indexzeile (gleicher Eintrag wie im Gruppenfeld des Maskeneditors). Der Parent-Eintrag gibt den Zugriffspfad auf den Vaterknoten des neuen Stichworts an. Die Wurzel wird dabei durch einen Punkt markiert. Ein Parenteintrag „.“ Erzeugt also ein Stichwort auf der untersten Ebene, „.AA“ erzeugt unterhalb des ersten Eintrags, „.AB“ unterhalb des zweitern Eintrags.

AnsiString AddSw(AnsiString Gruppe, AnsiString Parent, AnsiString Wort)

Parameter:

Gruppe	Wählt die Stichwortliste aus
Parent	Vorgängerknoten für den neuen Eintrag
Wort	Neues Stichwort

Rückgabewerte:

-1: kein Workspace offen
-2: Fehler beim Speichern des Stichwortes
sonst: Position des neuen Stichworts

Beispiel

```
...
Set Elo=CreateObject("ELO.office")

call Elo.DeleteSwl( "THM", "." )

MsgBox Elo.AddSw( "THM", ".", "1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.2" )
MsgBox Elo.AddSw( "THM", ".AA", "1.3" )
MsgBox Elo.AddSw( "THM", ".", "2" )
MsgBox Elo.AddSw( "THM", ".", "3" )

MsgBox Elo.ReadSwl( "THM", ".", " - " )
MsgBox Elo.ReadSwl( "THM", ".AA", " - " )

call Elo.UpdateSw( "THM", ".AB", "2a" )
MsgBox Elo.ReadSwl( "THM", ".", " - " )
...
```

Funktion AddThesaurus

Diese Funktion erzeugt einen Thesauruseintrag in der Datenbank. Jede Thesaurusgruppe muss eine eindeutige Gruppennummer (GroupId) besitzen, wenn Sie beim ersten Eintrag der Gruppe eine 0 übergeben, dann ermittelt ELO eine neue zufällige Nummer. Der Prio Parameter bestimmt die Sortierreihenfolge, der ListId Parameter muss im Augenblick fest auf 1 eingestellt werden (Thesaurus im Verschlagwortungsdialog).

int AddThesaurus(int ListId, int GroupId, int Prio, AnsiString Value)

Parameter:

ListId	Bereich, für die Verschlagwortung auf 1 setzen
GroupId	Thesaurusgruppe, 0: neue Gruppe ermitteln
Prio	Sortierreihenfolge
Value	Text

Rückgabewerte:

>0: ok, GroupId
-1: kein Workspace offen
-2: Fehler beim Speichern der Daten

Beispiel

```
Set Elo=CreateObject("ELO.office")

Group = Elo.AddThesaurus( 1, 0, 10, "Werkzeug" )
MsgBox Group
if Group > 0 then
  MsgBox Elo.AddThesaurus( 1, Group, 20, "Hammer" )
  MsgBox Elo.AddThesaurus( 1, Group, 30, "Schraubenzieher" )
  MsgBox Elo.AddThesaurus( 1, Group, 40, "Zange" )
end if
```

Funktion AnalyzeFile (invalid)

Diese Funktion sendet den Inhalt einer Postboxdatei an das OCR Modul und füllt die erkannten Bereiche in vordefinierte Maskenfelder. Der Name der Postboxdatei kann über SourceFile übergeben werden. Alternativ hierzu kann in diesem Parameter eine Zeilennummer (#0, #1, #2 ...) übergeben werden. Als Dateiname wird dann die entsprechende Datei aus der Postliste verwendet. Nach der Erkennung wird das Ergebnis in der Schlagwortdatei zu der Bilddatei abgespeichert.

Der Parameter OcrDescriptor enthält die Liste der zu untersuchenden Rechtecke und die zu füllenden Maskenfelder. Eine ausführliche Beschreibung dieser Liste finden Sie in der Barcode-Dokumentation.
Das Rechteck muß im Unterschied zum Barcode-Modul so definiert werden:
"R(Links,Oben,Rechts,Unten)"

int AnalyzeFile(AnsiString SourceFile, AnsiString OcrDescriptor, int MaskNo)

Parameter:

SourceFile	Zu untersuchende Datei oder Postlisten-Zeilennummer (mit #Nummer).
OcrDescriptor	Rechteckliste, im Barcode-Format.
MaskNo	Dokumententyp der zu erzeugenden Schlagwortdatei

Rückgabewerte:

1: ok
-1: kein Workspace offen
-2: OCR Subsystem nicht geladen
-3: keine Rechteckliste vorhanden
-4: fehlerhafte Rechteckliste
-5: Fehler bei der OCR Bearbeitung
-6: Fehler beim Schreiben der erkannten Daten

siehe auch: AddPostboxFile
ReadBarcodes

Property ArchiveDepth (int) (invalid)

Mit diesem Property können Sie die Anzahl der Hierarchiestufen des aktuellen Archivs ermitteln. Dabei zählt die Ebene der Dokumente mit, d.h. ein klassiches ELO-Archiv mit den Aktenstrukturelementen Schrank-Ordner-Register-Dokument hat 4 Hierarchiestufen.
Ist die Archivansicht noch nicht aktiv, hat das Property ArchiveDepth den Wert –1.

siehe auch:

Funktion ArcListLineId

Mit dieser Funktion kann die ELO Objektid einer Zeile aus der rechten Archivliste ermittelt werden.

int ArcListLineId(int LineNo)

Parameter:

LineNo	zu selektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
>0: ObjektId

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )

for i = 0 to 8
  res = res & Elo.ArcListLineSelected( i ) & " - " & Elo.ArcListLineId( i
) & ", "
next

for i = 0 to 3
  Elo.SelectArcListLine( i )
next

for i = 4 to 7
  Elo.UnselectArcListLine( i )
next

Elo.SelectArcListLine( 8 )

MsgBox res
```

siehe auch: UnselectArcListLine
SelectArcListLine
ArcListLineSelected

Funktion ArcListLineSelected

Mit dieser Funktion kann getestet werden, ob eine Zeile in der rechten Archivliste selektiert ist.

int ArcListLineSelected(int LineNo)

Parameter:

LineNo	zu selektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
0: Zeile nicht selektiert
1: Zeile selektiert

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )

for i = 0 to 8
  res = res & Elo.ArcListLineSelected( i ) & " - "
next

for i = 0 to 3
  Elo.SelectArcListLine( i )
next

for i = 4 to 7
  Elo.UnselectArcListLine( i )
next

Elo.SelectArcListLine( 8 )

MsgBox res
```

siehe auch: UnselectArcListLine
SelectArcListLine

Property AttId (int)

Das Property AttId bestimmt die Arbeitsversion des Dateianhangs eines Dokuments. Dieser Eintrag muss aus der Liste der vorhandenen Attachments entnommen werden, ein fremder Eintrag kann hier zu schweren Fehlfunktionen führen.

Beispiel:

```
Set ELO = CreateObject( „ELO.office“ )
MsgBox Elo.AttId
```

siehe auch: MaskKey
DocKey
DocKind
DocPath

Property AutoDlgResult (AnsiString)

Dieses Property übergibt das Resultat für alle Objekte des AutoDialoges. Die einzelnen Werte sind mit einem Return getrennt.

Bsp.:

```
Übergibt den Inhalt des Edit Feldes.
Elo.CreateAutoDlg (“Neuer Dialog“)
Elo.AddAutoDlgControl (4,1,“Name“,““)
Elo.ShowAutoDialog
MsgBox Elo.AutoDlgResult
```

siehe auch: CreateAutoDlg
AddAutoDlgControl
ShowAutoDlg
GetAutoDlgValue

Funktion BringToFront

Mit der Funktion BringToFront können Sie ELO auf dem Desktop in den Vordergrund bringen und damit für den Anwender sichtbar machen, wenn ELO durch ferngesteuerte andere Applikationen zuvor verdeckt wurde.

void BringToFront()

Parameter:
keine

Rückgabewerte:
keine

siehe auch: EloWindow

Funktion ChangeObjAcl (int ObjId, AnsiString Acl, int Option) (invalid)

Mit dieser Funktion können die Zugriffsrechte für die Objekte zugeteilt und/oder geändert werden.

AnsiString ChangeObjAcl (int ObjId, AnsiString Acl, int Option)

ObjId : Interne ELO Id

Acl : Zu setzende Rechte. Mehrere Rechte sind mit dem Komma zu trennen.
Format : XYZ
XY:

- RW – Schreibrecht und Leserecht
- RD – Leserecht
- WT – Schreibrecht
- KY – Schlüssel

Z :
Usernummer / Gruppennummer/ Keynummer

Option : 32 Bit Integermaske

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	04	03	02	01	00

00	0 – Rechte hinzufügen 1 – Rechte überschreiben
01	0 – Nicht untergeordnete Objekte weiterreichen 1 – An untergeordnete Objekte weiterreichen (nicht implementiert)
02	0 – Identische Rechte weiterreichen (nicht implementiert) 1 – Nur Veränderungen weiterreichen (nicht implementiert)
03	0 – Warndialog bei Eigenrechtentfernung ein 1 – Warndialog bei Eigenrechtentfernung aus
04	0 – 1 – Vorgängerrechte übernehmen (nicht implementiert)
x	Momentan unbenutzt (reserviert für Erweiterungen)

Rückgabewert:

- -1 - Kein Arbeitsbereich aktiv
- -2 - unzureichende Rechte um Änderungen vorzunehmen
- -3 - Benutzerabbruch bei Eigenrechtentfernung (Dialog)

Ansosnst Neue Objektrechte

Bsp.:

Zeigt die aktuellen Rechte von Objekt mit der Id 45 an.
Rechte = Elo.ChangeObjAcl (45,““,0)

Setzt für das Objekt 30 den Systemschlüssel und entfernt alle anderen
Elo.ChangeObjAcl (30,“KY0“,1)

siehe auch:

Funktion CheckFile

Mit der Funktion CheckFile können Sie verschiedene Dateieigenschaften abfragen.

Int CheckFile(int OptionNo, AnsiString Dateiname)

Parameter:

OptionNo:	0: Exklusiven Zugriff auf die Datei prüfen
Dateiname:	Name der zu prüfenden Datei

Rückgabewerte:

1:	Ok
-1:	Ungültige Nummer unter OptionNo
-2:	Exklusiver Zugriff nicht möglich

Verfügbar seit: 3.00.288

siehe auch:

Funktion CheckFileHash

Mit der Funktion CheckFileHash können Sie zu einer Datei prüfen lassen, ob diese schon im ELO abgelegt ist. Dem Anwender wird dann ein Dialog vorgelegt, in dem er entscheiden kann, ob er das Dokument trotzdem ablegen möchte, statt der Ablage lieber eine Referenz einträgt oder die Ablage komplett abbrechen möchte.

Int CheckFileHash(AnsiString HeaderMessage, AnsiString FileName, int ModeMask)

Parameter:

HeaderMessage:	Zusätzliche Anmerkung im Dialogtitel (z.B. Dateiname oder Bezeichnung)
FileName:	Name und Pfad der zu prüfenden Datei
ModeFlag:	Reserviert, muss auf 128 gesetzt werden.

Rückgabewerte:

0:	Dokument im Archiv noch nicht vorhanden
1:	Dokument vorhanden, trotzdem ablegen.
>1:	Dokument vorhanden, Referenz auf diese DocId bilden
-1:	Kein Arbeitsbereich aktiv, keine Kontrolle möglich
-2:	Dokument vorhanden, keine Ablage vornehmen

Beispiel:

```
Set Elo=CreateObject("ELO.office")
MsgBox Elo.CheckFileHash( "Testdatei", "d:\temp\Scandatei.tif", 128 )
```

siehe auch: GetMD5Hash
LookupHistMD5

Funktion CheckIn

Funktion CheckInEx

Über die Funktion können Sie ein Dokument, welches Sie zuvor per CheckOut aus dem Archiv erhalten haben, wieder einchecken. Beachten Sie bitte, daß Sie den Dateinamen nicht verändern dürfen, da dieser die Kennung für das verwendete Archiv und die Objektnummer enthält. Unter ELOoffice 6.0 führt der Befehl CheckIn zu einer Abfrage der Versionsnummer und eines Kommentars. Diese Abfrage können Sie durch Verwendung des Befehls CheckInEx unterdrücken, diese Angaben werden statt dessen als Parameter mitgegeben.

int CheckIn(AnsiString FileName)
int CheckInEx(AnsiString FileName, AnsiString sComment, AnsiString sVersion)

Parameter:

FileName	: Name der einzucheckenden Datei
sComment	: Kommentar zu der neuen Dokumentenversion
sVersion	: Versionsnummer

Rückgabewerte:

>0:	ObjektId des eingecheckten Dokuments
-100:	Kein Arbeitsbereich aktiv
-1..-10	CIO_LockError, CIO_PathError, CIO_CopyError, CIO_ExecError, CIO_UknError, CIO_NoDocument, CIO_ModeError, CIO_UserAbort, CIO_ArchiveError, CIO_ReadOnly

Beispiel:

```
function CryptIt( ObjId, Schluesselkreis )
  CryptIt=1
  x=Elo.PrepareObjectEx( ObjId, 0, 0 )
  if x<0 then
    CryptId=-1
  else
    if (Elo.ObjFlags and 256)=0 then
      Elo.ObjFlags=Elo.ObjFlags or (4096*Schluesselkreis) or 256
      Elo.UpdateObject
      FName=Elo.CheckOut(objid,0)
      if FName<>"" then
        if Elo.CheckInEx(FName,"Automatische Verschlüsselung","1.0")<0 then
          CryptId=-3 ' Fehler beim CheckIn
        end if
      else
        CryptId=-2 ' Fehler beim CheckOut
      end if
    else
      CryptIt=2 ' ist bereits verschlüsselt, nichts zu tun...
    end if
  end if
end function
```

Property CheckInOutFileName (AnsiString)

Mit Hilfe dieses Properties können Sie den Dateinamen des Dokuments innerhalb der Ereignisse „Beim Aus-/Einchecken eines Dokuments“ ermitteln. Beim Ereignis „Vor dem Einchecken“ kann der Dateiname bei Bedarf verändert werden.

siehe auch: CheckInOutObjID
ActionKey

Property CheckInOutObjID (int)

Mit Hilfe dieses Properties können Sie die Objekt-ID des Dokuments innerhalb der Ereignisse „Beim Aus-/Einchecken eines Dokuments“ ermitteln.

siehe auch: CheckInOutFileName
ActionKey

Funktion CheckObjAcl

Über die Funktion CheckObjAcl können Sie nachprüfen, was für Zugriffsberechtigungen Sie auf ein Objekt besitzen. Wenn das Objekt bereits aktiv ist (z.B. innerhalb eines Verschlagwortungsevents oder nach einem PrepareObjectEx), kann geben Sie als lObjectId eine 0 an. In diesem Fall wird der aktuelle Wert aus dem ObjAcl Property verwendet. Wenn das Objekt noch nicht aktiv ist, können Sie die ObjectId als Parameter mit angeben. Es wird dann die ACL Liste des Objekts geladen und geprüft. Da nur das eine Property und nicht das gesamte Objekt aus der Datenbank gelesen wird, ist dieser Weg schneller als die Kombination PrepareObjectEx(Id...) und CheckObjAcl(0). Das gilt natürlich nur dann, wenn nicht andere Stellen des Scripts ohnehin ein PrepareObjectEx erfordern.

AnsiString CheckObjAcl(int lObjectId)

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )
Id = Elo.GetEntryId(-1)

' Nur ObjectId bekannt:
MsgBox Elo.CheckObjAcl( Id )


if Elo.PrepareObjectEx( Id, 0, 0 ) then
  ' Das Objekt ist bereits geladen:
  MsgBox Elo.CheckObjAcl( 0 )
end if
```

Parameter:

lObjectId	Nummer des zu prüfenden Dokuments (interne ELO Id) oder 0

Rückgabe

-1	: Kein Arbeitsbereich aktiv
-2	: Fehler beim Lesen der ACL
>=0	: Berechtigungsmaske (1: Read, 2: Write, 4: Delete, 8: Edit Document)

Funktion CheckOut

Über die Funktion CheckOut können Sie ein Dokument zum Bearbeiten aus dem aktiven Archiv entnehmen. Dieses kann dann nach der Bearbeitung über die Funktion CheckIn wieder ins Archiv übertragen werden. Beachten Sie bitte, daß Sie den Dateinamen nicht verändern dürfen, da dieser die Kennung für das verwendete Archiv und die Objektnummer enthält.

AnsiString CheckOut(int lObjectId, int iMode)

Parameter:

lObjectId	Nummer des auzulesenden Dokuments (interne ELO Id)	
iMode	0	Nur auslesen, Applikation nicht aktivieren
	1	Auslesen und Applikation ohne Nachfrage aktivieren
	2	Auslesen und Applikation nach Bestätigung aktivieren

Rückgabewert:

Dateiname des ausgecheckten Dokumentes

Funktion CheckPage

Über die Funktion CheckPage können Sie zum aktuellen Postboxeintrag prüfen, ob eine Trenn- oder Leerseite vorliegt. Im Vorspann finden Sie ein Beispiel zu diesem Befehl

Int CheckOut(int Mode, int Trefferanteil)

Parameter:

Mode	1: Leerseitenkontrolle 2: Trennseitenkontrolle (auch Kombinationen zulässig)
Trefferanteil	0...999 benötigte Treffer in Promille. Ein Wert von 970 (default im ELO Client) verlangt eine Trefferrate von 97%.

Funktion CheckUpdate

Mit der Funktion können Sie entscheiden, ob eine Änderung sofort angezeigt wird. Besonders bei der automatischen Erfassung von Massendaten kann es viel Zeit sparen, wenn ELO nicht jede Änderung anzeigt. In diesem Fall setzen Sie vor dem Lauf ein ELO.CheckUpdate(0) und nach dem Lauf ein ELO.CheckUpdate(1) ein.

int CheckUpdate(int DoCheck)

Parameter:
DoCheck : 0: keine Anzeige, 1: Anzeige

Rückgabewerte:
Alter Zustand

siehe auch: EloWindow

Funktion ClickOn

Mit Hilfe dieser Funktion kann ein Mausklick auf ein Dialogelement im ELO Hauptdialog simuliert werden. Somit können alle Funktionen, die in den Menüs oder in der Toolbar zur Verfügung stehen, aufgerufen werden.

int ClickOn(AnsiString ComponentName)

Parameter:

ComponentName: Name des Dialogelements (Liste aller Dialogelement s. Anhang A)

Rückgabewerte:

-1	Hauptdialog nicht aktiv
-2	Komponente nicht vorhanden
-3	Komponente besitzt keine OnClick-Routine
-4	Komponententyp wird nicht unterstützt
-5	Funktion derzeit nicht aufrufbar (Dialogelement „disabled“)
1	Ok

siehe auch:

Funktion CloseActivateDocDlg (invalid)

Mit der Funktion CloseActivateDocDlg können Sie ein Dokument, welches aus Elo heraus zum Bearbeiten aktiviert wurde wieder im Elo einchecken. Hierzu haben Sie die Optionen Mode=1: Ok, neue Version übernehmen und Mode=2: Abbruch: Änderungen verwerfen zur Verfügung

int CloseActivateDocDlg(int Mode)

Parameter:

Mode	1: Ok, 2: Abbruch

Rückgabewerte:

-1	ActivateDocDlg nicht aktiv
-2	falscher Mode-Parameter
-3	fehler beim Schließen des Dialogs
1	Ok

siehe auch:

Funktion CollectChildList

Über die Funktion CollectChildList können Sie zu einer Objekt-Id alle Nachfolgerknoten ermitteln. Die Knotenliste wird als Textstring übergeben, die einzelnen Nachfolger-Ids sind durch Doppelpunkte getrennt.

AnsiString CollectChildList(int ObjId)

Parameter:

ObjId Knotennummer des Startobjekts

Rückgabewerte:

leer: Fehler, nicht näher spezifiziert
ansonsten: Nachfolgerliste

Funktion CollectLinks

Über die Funktion CollectLinks können Sie alle Links zu einer Objekt-Id ermitteln. Die Ids der verlinkten Objekte werden als Kommaliste zurückgegeben.

AnsiString CollectLinks(int iObjID, int iMode)

Parameter:

iObjId	Objekt-Id des ELO Objekts
iMode	0

Rückgabewerte:

„-1“	Kein Arbeitsbereich aktiv
leer:	keine Links vorhanden
ansonsten:	Nachfolgerliste (Objekt-Ids durch Komma getrennt)

Funktion CollectWv

Über die Funktion CollectWv können Sie die Wiedervorlagetermine eines Anwenders sammeln. Diese Liste wird nicht im ELO angezeigt, sie kann nur über die Funktion GetListEntry ausgelesen werden.

Achtung: die temporäre interne Liste wird auch von DoInvisibleSearch verwendet. Wenn Sie beide Aktionen in einem Script verwenden, müssen Sie vor einem Wechsel jeweils die komplette Ergebnisliste auslesen, ansonsten werden die Resultate verworfen.

int CollectWv(int UserId, AnsiString StartDate, AnsiString EndDate, int Prio)

Parameter:

UserId	Anwendernummer (plus 0x40000000 für Gruppen und 0x20000000 für Vertretungen)
StartDate	Anfangsdatum des zu sammelnden Bereichs
EndDate	Endedatum des Sammelbereichs
Prio	Priorität (nur bei Wiedervorlagen und Aktivitäten (1..3))

Rückgabewerte:

-1:	Kein Arbeitsbereich aktiv
-2:	Ungültiges Datum
>=0:	Anzahl der gefundenen Einträge

Beispiel:

```
Set Elo=CreateObject("ELO.office")
UserId=Elo.ActiveUserId ' nur die eigenen Termine
'UserId=Elo.ActiveUserId + 1610612736 ' alle Termine

cnt=Elo.CollectWv( UserId, "22.06.2006", "30.06.2006", 3)
for i=0 to cnt-1
  msg=msg & Elo.GetListEntry(i) & vbcrlf
next
MsgBox msg
```

siehe auch: GetListEntry

Property ColorInfo (int)

Das Property ColorInfo bestimmt den Farbwert einer Farbdefinition. Dieser Farbwert wird in der Windowsüblichen RGB Schreibweise übergeben.

siehe auch: ReadColorInfo
WriteColorInfo
ColorName

Property ColorName (AnsiString)

Das Property ColorName bestimmt den Farbnamen (Kurzbezeichnung) einer Farbdefinition.

Maximale Länge: 30 Zeichen

siehe auch: ReadColorInfo
WriteColorInfo
ColorInfo

Property ColorNo (int)

Das Property ColorNo bestimmt die Farbnummer einer Farbdefinition.

siehe auch: ReadColorInfo
WriteColorInfo
ColorName

Funktion CreateAutoDlg (AnsiString Caption)

Erstellt einen leeren Dialog mit einem Ok & Abbrechen Button.

int CreateAutoDlg (AnsiString Caption)

Caption :
Titel des Dialogs

Rückgabewert :
1 – Dialog konnte erstellt werden.

Bsp.:

```
Ein leerer Dialog wird angezeigt.
Elo.CreateAutoDlg ("Ein leerer Dialog")
Elo.ShowAutoDlg
```

siehe auch: AddAutoDlgControl
ShowAutoDlg
GetAutoDlgValue

Funktion CreateStructure (AnsiString Path, int StartID)

Diese Funktion erstellt eine Liste mit Schrank, Ordner, Registern, usw. ab momentaner Position oder wenn am Anfang ein Separator steht ab Schrankposition. Bestehende gleichlautende Pfade werden nur dann neu Initialisiert, wenn am Ende des Pfades ein Separator steht.

(AnsiString) CreateStructure (AnsiString Path, int StartID)

Path : Zu erstellender Pfad
Einzelne Elemente sind mit dem Separator zu trennen
Separator am Anfang Ab Schrankposition erstellen
StartID wird ignoriert
Separator am Ende Existierende Pfade werden überschrieben

StartID : Gültige Werte : ObjektID , 0 , negative ObjektID
0 – Erstellung des Pfades innerhalb des augenblicklich ausgewählten Objektes
Eine negative StartID bewirkt die Erstellung innerhalb des Objektes eine
Positive auf gleicher Ebene.

Der Separator am Anfang eines Pfades hat Vorrang vor der StartID. (StartID wird ignoriert)

Rückgabewerte :
Liste von ObjektIDs der erstellten Elemente getrennt durch den Separator (235¶252¶22)

-5 – Kein Eintrag ausgewählt
-4 – Leerer Feldname
-3 – Keine gültige Startposition angegeben
-2 – Kein Arbeitsbereich aktiv
-1 – Zu viele Objekte (Archivtiefe wird überschritten)

Bsp.:

Erstellt einen Schrank, Ordner und Register. Startposition ist die Aktenschrankebene
ELO. CreateStructure („¶Mein_Schrank¶Mein_Ordner¶Mein_Register”,0)

Erstellt Register mit Unterregister oder Ordner mit Register oder Schrank mit Ordner
Abhängig von der aktuellen Position im Archiv
ELO. CreateStructure („123¶222“,0)

Erstellt Register mit Unterregister oder Ordner mit Register oder Schrank mit Ordner
Auf gleicher Ebene des angegebenen Objektes
ELO. CreateStructure („123¶222“,2345)

Erstellt Register mit Unterregister oder Ordner mit Register oder Schrank mit Ordner
Innerhalb des angegebenen Objektes
ELO. CreateStructure („123¶222“,-764)

siehe auch:

Funktion CreateViewer

Über die Funktion können Sie aus einem Exportdatensatz einen Viewerdatensatz erstellen lassen. Hierzu muss das ViewerPostbox Verzeichnis unter ELOoffice vorbereitet sein und ein fertiger Exportdatensatz vorliegen. Der Zielpfad darf bereits einen Viewer enthalten. Wenn es das Zielarchiv bereits gibt, werden die Datensätze in dieses Archiv eingefügt. Im Zielbereich dürfen bis zu 4 Archive angelegt werden.

Int CreateViewer(AnsiString ExportPath, AnsiString ArcName,AnsiString DestPath)

Beispiel:

```
Set Elo=CreateObject("ELO.office")
call Elo.CreateViewer( "D:\ExportPfad", "test1", "D:\Zielpfad" )
```

Parameter:

ExportPath	Verzeichnis, welches den Exportdatensatz enthält
ArcName	Viewer-Archivname
DestPath	Zielverzeichnis für den Viewer

Rückgabewerte:

-1 :	kein Arbeitsbereich aktiv
-2:	Zielpfad oder Archivname fehlen
-3:	Verzeichnisstruktur im Zielpfad konnte nicht erzeugt werden
-4:	Postboxverzeichnis im Zielpfad konnte nicht erzeugt werden
-5:	Viewerdateien konnten nicht kopiert werden
1:	ok

Funktion DateToInt(AnsiString Datum)

Diese Funktion wandelt einen Datumstext in das ELO-Interne numerische Datumsformat.

int DateToInt(AnsiString Datum)

Parameter:

Text	Zu wandelndes Datum.

Rückgabewerte:

0:	Ungültiger Datumseintrag
>0:	ELO Datumswert

siehe auch: IntToDate

Funktion DebugOut

Diese Funktion sendet einen Text an das Elo-Debug Fenster.

void DebugOut(AnsiString Text)

Parameter:

Text Auszugebender Text, wird mit Uhrzeit zusammen an das Debug Fenster geschickt.

Rückgabewerte:

Keine

siehe auch:

Funktion DeleteMask

Löscht eine ELO Verschlagwortungsmaske. Es wird vor der Löschoperation nachgeprüft, ob es noch Einträge mit dieser Maske im Archiv oder den gelöschten und noch nicht dauerhaft entfernten Objekten gibt. Falls noch Einträge vorhanden sind, wird der Befehl mit einem Fehler abgebrochen und in dem Property TextParam ist eine Liste der ersten 16 Objekt-Ids welche diese Maske verwenden.

int DeleteMask(int MaskNo)

Parameter:

MaskNo: Nummer der zu löschenden Maske

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Es gibt noch Einträge zu dieser Maskennummer
-3: Datenbankfehler beim Löschen
1: ok

Beispiel:

```
Set ELO = CreateObject("ELO.office")

MaskNo = ELO.LookupMaskName( "DieseMaskeWirdGelöscht" )
if MaskNo > 0 then
  Res = ELO.DeleteMask( MaskNo )
  if Res > 0 then
    MsgBox "Gelöscht"
  else
    MsgBox "Maske konnte nicht gelöscht werden, Fehlernummer: " & Res & ",
Objekte: " & ELO.TextParam
  end if
else
  MsgBox "Maske nicht gefunden"
end if
```

siehe auch: DeleteObj

Funktion DeleteObj

Löscht einen Elo Eintrag, vorgegeben durch die ObjektId. Falls es sich um einen Schrank, Ordner oder Register handelt, werden auch alle Untereinträge gelöscht.

int DeleteObj(int ObjektId)

Parameter:

ObjektId: Nummer des zu löschenden Eintrags

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehler beim Löschen
-3: Unzureichende Berechtigung zum Löschen
1: ok

siehe auch: PrepareObj
UpdateObj

Funktion DeleteSwl

Diese Funktion löscht einen Teilbaum aus einer Stichwortliste. Jeder Eintrag einer Stichwortliste wird innerhalb seiner Ebene eindeutig durch eine 2-stellige Buchstabenkombination AA, AB, AC .. ZZ gekennzeichnet. Durch die Verkettung aller Kennzeichen aller Ebenen können Sie jedes Stichwort im Baum eindeutig ansprechen. Beginnend mit dem Punkt für die Wurzel können Sie nun z.B. über „.AAABAC“ in der untersten Ebene den ersten Eintrag (AA), darunter den zweiten (AB) und darunter den dritten (AC) Eintrag löschen. Dabei wird der gewählte Eintrag mit allen evtl. vorhandenen Untereinträgen entfernt.

Die Gruppe kennzeichnet die Listenzuordnung zu einer Indexzeile (gleicher Eintrag wie im Gruppenfeld des Maskeneditors).

int DeleteSwl(AnsiString Gruppe, AnsiString Parent)

Parameter:

Gruppe	Wählt die Stichwortliste aus
Parent	Pfad auf die zu löschenden Einträge

Rückgabewerte:

-1: kein Workspace offen
-2: Fehler beim Speichern des Stichwortes
-3: Ungültiger Parent-Pfad
1: Ok

Beispiel

```
...
Set Elo=CreateObject("ELO.office")

call Elo.DeleteSwl( "THM", "." )

MsgBox Elo.AddSw( "THM", ".", "1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.2" )
MsgBox Elo.AddSw( "THM", ".AA", "1.3" )
MsgBox Elo.AddSw( "THM", ".", "2" )
MsgBox Elo.AddSw( "THM", ".", "3" )

MsgBox Elo.ReadSwl( "THM", ".", " - " )
MsgBox Elo.ReadSwl( "THM", ".AA", " - " )

call Elo.UpdateSw( "THM", ".AB", "2a" )
MsgBox Elo.ReadSwl( "THM", ".", " - " )
...
```

Funktion DeleteWv

Löscht einen Wiedervorlagetermin (bestimmt durch den Parameter WvIdent). Beim Löschen des Termins muß der Eigentümer mit angegeben werden, falls Sie hier eine –1 einsetzen wird Ihre eigene EloUserId verwendet.

int DeleteWv(int WvId, int UserOwner)

Parameter:

WvId: Nummer des zu löschenden Termins
UserOwner: Eigentümer des Termins

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehler beim Löschen
1: ok

Siehe auch: ReadWv
WriteWv
WvIdent
WvParent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Funktion DeleteWvLine

Mit Hilfe dieser Funktion läßt sich ein Eintrag innerhalb der Anzeige der Wiedervorlagetermine ausblenden. Der zugehörige Wiedervorlageeintrag wird nicht gelöscht.

int DeleteWvLine(int LineNo)

Parameter:

LineNo: Zeilennummer des auszublendenden Eintrags

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
0: Falsche Zeilennummer
1: ok

Siehe auch:
WriteWv
DeleteWv
WvIdent
WvParent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Property DelOutlookName (AnsiString)

Verfügbar ab:	

Dieses Property gibt den Namen der zu löschendenden Person (Gruppe) der Wiedervorlage an. Es wird benutzt wenn der Name der Person (Gruppe), an den die Wiedervorlage gerichtet ist, in ELO geändert wird damit der Eintrag entfernt werden kann.

siehe auch: OutlookName

Funktion DoCheckInOut

Diese Funktion öffnet einen Dialog zum Ein- und Auschecken von Dokumenten.

int DoCheckInOut (int hwndParent, AnsiString DlgTitel, AnsiString Short, AnsiString Desc, AnsiString XDate, AnsiString FileName,int Ctrl, int Minimize)

Parameter:

hwndParent	Windows-Fensterhandle des übergeordneten Fensters
DlgTitle	Titel des zu öffnenden Dialogs.
Short	Kurzbezeichnung (für neues Dokument einchecken)
Desc	Zusatztext (für neues Dokument einchecken)
XDate	Dokumentendatum (für neues Dokument einchecken)
FileName	Dateiname der Datei
Ctrl	-1 neues Dokument 0 Check out >0 Check in
Minimize	<>0 für ELO-Client minimieren

Rückgabewerte:

-1	kein Arbeitsbereich aktiv **oder** Ablage in Postbox
0	Abbruch
>0	Objekt Id

siehe auch: DoCheckInOut2

Funktion DoCheckInOut2

Diese Funktion öffnet einen Dialog zum Ein- und Auschecken von Dokumenten.

int DoCheckInOut 2(int hwndParent,
AnsiString sDlgTitle,
AnsiString sShort,
AnsiString sDesc,
AnsiString sXDate,
AnsiString sFilterExt,
AnsiString& sFileName,
int nMinimize)

Parameter:

hwndParent:	Windows-Fensterhandle des übergeordneten Fensters
sDlgTitle:	Titel des zu öffnenden Dialogs.
sShort:	Kurzbezeichnung (für neues Dokument einchecken)
sDesc:	Zusatztext (für neues Dokument einchecken)
sXDate:	Dokumentendatum (für neues Dokument einchecken)
sFilterExt:	Dateiendungen, die zur Auswahl stehen. Formatbsp: „Word Dokumente (*.doc;*.dot;*.txt)|*.doc;*.dot;*.txt|Excel Dokumente (*.xxx)|*.xls;*.txt“. Formateintrag für „Alle Dokumente|*:*“ wird automatisch ergänzt
nMinimize: <>0 für ELO-Client minimieren	

Rückgabewerte:

0 ... Abbruch
1 ... Neues Dokument gespeichert
2 ... Dokument eingecheckt
3 ... Dokument aus dem Archiv ausgecheckt
4 ... bereits ausgechecktes Dokument bearbeiten

Funktion DoCheckInOut3

Diese Funktion öffnet einen Dialog zum Ein- und Auschecken von Dokumenten.

int DoCheckInOut 3(int hwndParent,
AnsiString sDlgTitle,
AnsiString sShort,
AnsiString sDesc,
AnsiString sXDate,
AnsiString sFilterExt,
AnsiString& sFileName,
int nMinimize,
int iFlags,
AnsiString sInfo)

Parameter:

hwndParent:	Windows-Fensterhandle des übergeordneten Fensters
sDlgTitle:	Titel des zu öffnenden Dialogs.
sShort:	Kurzbezeichnung (für neues Dokument einchecken)
sDesc:	Zusatztext (für neues Dokument einchecken)
sXDate:	Dokumentendatum (für neues Dokument einchecken)
sFilterExt:	Dateiendungen, die zur Auswahl stehen. Formatbsp: „Word Dokumente (*.doc;*.dot;*.txt)|*.doc;*.dot;*.txt|Excel Dokumente (*.xxx)|*.xls;*.txt“. Formateintrag für „Alle Dokumente|*:*“ wird automatisch ergänzt
nMinimize:	<>0 für ELO-Client minimieren
iFlags:	Bit 0 gesetzt: Verschlagwortungsinformationen werden aus internem, zuvor mit PrepareObjectEx initialisierten Objekt übernommen Bit 0 nicht gesetzt: Verschlagwortungsinformationen werden den Parametern sShort, sDesc und sXDate entnommen
sInfo:	reserviert für spätere Erweiterungen

Rückgabewerte:

0 ... Abbruch
1 ... Neues Dokument gespeichert
2 ... Dokument eingecheckt
3 ... Dokument aus dem Archiv ausgecheckt
4 ... bereits ausgechecktes Dokument bearbeiten

Property DocId (int)

Das Property DocId bestimmt die Arbeitsversion eines Dokuments. Dieser Eintrag muss aus der Liste aller Dokumentendateien aus der Versionsgeschichte dieses Dokuments ausgewählt worden sein. Ein fremder Eintrag an dieser Stelle kann zu schweren Fehlfunktionen führen.

siehe auch: MaskKey
DocKey
DocKind
DocPath

Property DocKey (int) (invalid)

Das Property DocKey bestimmt den Schlüssel eines Eintrags. Ein Eintrag kann ein Schrank, Ordner, Register oder Dokument sein.

Wenn Sie eine Maskendefinition bearbeiten, enthält dieser Eintrag die Standardvorgabe für neue Dokumente dieses Typs (der MaskKey enhält den Schlüssel für die Maske selber).

siehe auch: MaskKey
DocKey
DocKind
DocPath

Property DocKind (int)

Das Property DocKind bestimmt die Farbe eines Eintrags. Ein Eintrag kann ein Schrank, Ordner, Register oder Dokument sein.

Wenn Sie eine Maskendefinition bearbeiten, enthält dieser Eintrag die Standardvorgabe für neue Dokumente dieses Typs.

Die Farbnummer entspricht der Zeilennnummer aus dem Dialog "Systemverwaltung -> Schriftfarbe... -> Farbverwaltung"

siehe auch: DocKey
DocPath

Property DocPath (int)

Das Property DocPath bestimmt den Ablagepfad eines Eintrags. Ein Eintrag kann ein Ordner oder Dokument sein.

Wenn der Eintrag ein Ordner ist, kann dieser Pfad bei entsprechender Systemeinstellung als Vorgabewerte für Dokumente in diesem Ordner sein. Diese Betriebsart stellt allerdings nur ein Kompatibilitätsmodus für alte ELO office Archive dar, für neue Archive sollte sie nicht mehr verwendet werden.

Wenn Sie eine Maskendefinition bearbeiten, enthält dieser Eintrag die Standardvorgabe für neue Dokumente dieses Typs.

siehe auch: DocKey
DocKind

Property DocTPath (int)

Das Property DocTPath bestimmt den Vorgabe-Ablagepfad eines Eintrags in einer Maskendefinition.

siehe auch: DopPath

Funktion DoEditObjectEx

Über diese Funktion können Sie den Verschlagwortungsdialog aufrufen. Er zeigt die Daten an, die vorher mittels PrepareObjectEx initialisiert worden sind. Wenn die Daten aus einen vorhandenen ELO Objekt kommen dann kommt es hier zu Konflikten mit der automatischen Anzeigeaktualisierung, welche dazu führt, dass die OLE Eingabedaten wieder von den Originaldaten überschrieben werden. Aus diesem Grund muss in diesem Fall der Aufruf in eine CheckUpdate(0) – CheckUpdate(1) Klammer gepackt werden.

Der Aufruf von DoEditObject entspricht einem DoEditObjectEx(1,-1,0)

int DoEditObjectEx(int Mode, int Handle, int Minimize)

Parameter:

Mode: 1 – Edit, 2 – Search
Handle: Handle des Parent-Windows, Standard = -1
Minimize: 1-ELO Hauptfenster minimieren (nur Edit-Dialog sichtbar), 0-nicht minimieren

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1 : Dialog mit Ok beendet
2 : Dialog mit Abbruch beendet

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )

sELOObjID = Elo.GetEntryId(-1)
Rsp = Elo.PrepareObjectEx(sELOObjID, 0, 0)
Elo.CheckUpdate(0)
Rsp = Elo.DoEditObject()
Elo.CheckUpdate(1)
Rsp = Elo.UpdateObject
```

Siehe auch:

Funktion DoExecute

Mit Hilfe dieser Funktion kann eine andere Applikation gestartet werden, intern wird die Windows API-Funktion *ShellExecute* aufgerufen.
Variante 1: im Parameter FileName wird der Name eines Programms (EXE-Datei) übergeben.
Variante 2: im Parameter FileName wird der Name einer Datei übergeben, die mit diesem Dateityp verknüpfte Applikation wird gestartet.

int DoExecute(AnsiString FileName)

Parameter:

FileName	zu öffnende / auszuführende Datei

Rückgabewerte:

<=32: Windows Fehlercode der Funktion *ShellExecute*
>32 : Erfolg

Siehe auch: RunEloScript
DoExecuteEx

Funktion DoExecuteEx

Mit Hilfe dieser Funktion kann eine andere Applikation gestartet werden, intern wird die Windows API-Funktion *ShellExecute* aufgerufen.
Variante 1: im Parameter FileName wird der Name eines Programms (EXE-Datei) übergeben.
Variante 2: im Parameter FileName wird der Name einer Datei übergeben, die mit diesem Dateityp verknüpfte Applikation wird gestartet.

int DoExecuteEx(AnsiString File, AnsiString Param, AnsiString Verzeichnis, AnsiString Action, int Mode)

Parameter:

File	zu öffnende / auszuführende Datei
Param	zusätzliche Parameter (kann leer bleiben)
Verzeichnis	Startverzeichnis (kann leer bleiben)
Action	„open“, „print“ oder „explore”, (kann leer bleiben, es wird dann ein “open” ausgeführt)
Mode	Fenstergröße beim Start (-1: SW_SHOWNORMAL). Die hier möglichen Konstanten können in der Windows Hilfe unter dem Thema „ShellExecute“ nachgelesen werden.

Rückgabewerte:

<=32: Windows Fehlercode der Funktion *ShellExecute*
>32 : Erfolg

Siehe auch: RunEloScript
DoExecute

Funktion DoFullTextSearch

Die Funktion DoFullTextSearch startet eine Volltextsuche. Als Ergebnis der Funktion wird die Anzahl der gefundenen Referenzen zurückgegeben, diese stehen dann in der Rechercheliste zur Verfügung und können über SelectLine zur Anzeige gebracht werden.

int DoFullTextSearch(AnsiString sSearchString, int iSearchOption)

Parameter:

sSearchString	zu suchender Text
iSearchOption	0=UND-ODER-verknüpfte Suchbegriffe 1=intuitive Suche

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehler bei der Volltextsuche
ansonsten: Anzahl der gefundenen Einträge.

Siehe auch: DoSearch
SelectLine

Funktion DoInvisibleSearch

Die Funktion DoInvisibleSearch startet einen unsichtbaren Suchvorgang aus den ObjShort, ObjMemo, ObjAttrib Einträgen. Hierzu wird zuerst ein neuer Objekt-Datensatz mit PrepareObject angelegt, dieser Datensatz über die Zugriffsoperationen mit den zu suchenden Begriffen gefüllt und die Funktion DoInvisibleSearch aufgerufen. Als Ergebnis der Funktion wird die Anzahl der gefundenen Referenzen zurückgegeben, diese stehen dann in einer unsichtbaren internen Rechercheliste zur Verfügung und können über GetEntryId() abgefragt werden. Hierzu muss der Funktion GetEntryId allerdings eine besondere Zeilennummer übergeben werden, damit sie erkennen kann, dass nicht die normale Liste sondern die unsichtbare Liste abgefragt werden soll. Diese Nummer setzt sich zusammen aus der Zeilennummer und einem konstanten Faktor von 268435456 (das ist 0x10000000).

Beispiel

```
set Elo = CreateObject("ELO.office")

'suche alle Rechnungseinträge
MaskNo=Elo.LookupMaskName("Rechnung")
Elo.PrepareObjectEx 0,0,MaskNo

Elo.DoInvisibleSearch

for i=0 to 1000
  id=Elo.GetEntryId(268435456+i)
  if id<2 then exit for
  Elo.PrepareObjectEx id,0,0
  Result=Result & Elo.ObjShort & vbCrLf
next

MsgBox Result
```

int DoInvisibleSearch()

Parameter:

Keine

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
ansonsten: Anzahl der gefundenen Einträge.

Siehe auch: PrepareObject
SelectLine
DoFulltextSearch

Funktion DoSearch

Funktion DoSearchSel(AnsiString)

Funktion DoSearchEx(AnsiString, int)

Die Funktion DoSearch startet einen Suchvorgang aus den ObjShort, ObjMemo, ObjAttrib Einträgen. Hierzu wird zuerst ein neuer Objekt-Datensatz mit PrepareObject angelegt, dieser Datensatz über die Zugriffsoperationen mit den zu suchenden Begriffen gefüllt und die Funktion DoSearch aufgerufen. Als Ergebnis der Funktion wird die Anzahl der gefundenen Referenzen zurückgegeben, diese stehen dann in der Rechercheliste zur Verfügung und können über SelectLine zur Anzeige gebracht werden.

Es steht noch ein spezieller Modus zur Verfügung, der dazu dient, eine Liste von Elo-Objekten anzeigen zu lassen. In diesem Fall wird ein Maskeninhalt vom Typ „Freie Eingabe“ so gefüllt, daß nur das Feld mit der Kurzbezeichnung mit einem String der Form „¶Id1¶Id2¶Id3¶ ... ¶IdN¶“ gefüllt ist. In der Rechercheansicht wird dann eine Liste mit diesen Objekten angezeigt.

Über den Mode-Parameter können Sie festlegen, ob vor einer Suche ein bereits vorhandenes Suchergebnis gelöscht werden soll. Hierüber können Sie dann mehrfach-Suchen zusammenstellen. Bei der ersten Suche wird gelöscht, alle folgenden Suchen ergänzen dann die bislang gesammelte Liste.

int DoSearch()
int DoSearchSel(AnsiString SelectObject)
int DoSearchEx(AnsiString SelectObject, int Mode)

Parameter:

SelectObject	Falls im Rechercheergebnis ein Eintrag mit der vorgegebenen Kurzbezeichnung existiert, dann wird diese Zeile selektiert.
Mode	Bit 0: 0 alten Listeninhalt nicht löschen 1: vor der Recherche löschen Bit1..31: reserviert.

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
ansonsten: Anzahl der gefundenen Einträge.

Verfügbar seit:

DoSearchEx: 3.00.358

Beispiel:

```
' führt drei Suchen nach den Begriffen ELO, Test und Word
' aus und zeigt das Ergebnis in einer Trefferliste an
Set Elo=CreateObject( "ELO.office" )

call Elo.PrepareObjectEx(0,0,0)
Elo.ObjShort="elo"
call Elo.DoSearchEx( "", 1 )
```

```
call Elo.PrepareObjectEx(0,0,0)
Elo.ObjShort="test"
call Elo.DoSearchEx( "", 0 )

call Elo.PrepareObjectEx(0,0,0)
Elo.ObjShort="Word"
call Elo.DoSearchEx( "", 0 )
```

Siehe auch: PrepareObject
SelectLine
DoFulltextSearch

Funktion DoSelArcTree

Öffnet einen Dialog zur Auswahl eines ELO-Eintrags.

int DoSelArcTree(int hwndParent,
AnsiString sDlgTitle,
int nCtrl,
int nMinimize);

Parameter:
HwndParent: Windows-Handle des übergeordneten Fensters oder NULL
sDlgTitle: Dialogtitel
nCtrl: 1 für nur Register können ausgewählt werden
nMinimize: <> 0 für ELO minimieren.

Rückgabewerte:
-1: für Fehler,
0: für Abbrechen,
>0: Objekt-ID des ausgewählten Eintrags

Property EditDlgActive (int)

Dieses Property gibt an ob ein Dialog von ELO aus geöffnet ist.

Rückgabewerte:

0 – Nein
1 – Ja

siehe auch:

Funktion EditWv (int WvId, int ParentId)

Diese Funktion bearbeitet einen Wiedervorlagetermin.

int EditWv (int WvId, int ParentId)

WvId :	Interne ELO Id zu dem Termin
ParentId :	Id des verknüpften ELO-Objekts

Rückgabewert:

-1	Kein Arbeitsbereich aktiv
0	Dialog mit Abbruch verlassen
1	Dialog mit Ok verlassen

Bsp.:

siehe auch: WVNew
WvDueDate
WvListInvalidate
WvActionCode

Funktion EloWindow

Über diese Funktion kann bestimmt werden, wie die ELO Arbeitsoberflächen angezeigt werden. Als Parameter kann ‚MINIMIZE' mitgegeben werden, in diesem Fall wird keine Arbeitsoberfläche sichtbar, ‚MAXIMIZE', die sichtbaren Arbeitsoberflächen belegen den vollen Bildschirm oder ‚NORMAL', d.h. es werden die Standardeinstellungen für die Größe der Arbeitsoberfläche verwendet.

int EloWindow(AnsiString DisplayMode)

Parameter:

DisplayMode ‚MINIMIZE', ‚NORMAL' oder ‚MAXIMIZE'

Rückgabewerte:

0,1,2: Anzahl der offenen Arbeitsbereiche

siehe auch: BringToFront

Funktion Export

Über die Funktion Export können Sie ein (Teil-)Archiv exportieren.
Die Kennummern für die verfügbaren Export-Optionen sind in folgender Liste:

```
// Dokumente aus der Aktenstruktur nicht mitkopieren
#define IEX_SUPPRESSDOCS       (1 << 0)
// Chaos-Docs mitkopieren
#define IEX_CHAOSDOCS        (1 << 1)
// Kopierte Dokumente nach Abschluß löschen
#define IEX_ERASEDOCS          (1 << 2)
// Beim Importieren neue Aktenstruktur erstellen
#define IEX_NEWHIRARCHY        (1 << 3)
// Ablagedatum statt Dokumentendatum verwenden
#define IEX_USEIDATE           (1 << 4)
// Volle Lageinformation mitgeben
#define IEX_PATHINFO           (1 << 5)
// Archiv-Eintrag exportieren
#define IEX_ARCHIVEENTRY       (1 << 7)
#define IEX_TXT_ARCHIVEENTRY "Archive Entry"
// Klemmbrett exportieren
#define IEX_CLIPBOARD          (1 << 8)
#define IEX_TXT_CLIPBOARD      "Clipboard"
// Suchen exportieren
#define IEX_SEARCHLIST         (1 << 9)
#define IEX_TXT_SEARCHLIST     "Searchlist"
// Verschlüsselte Dokumente exportieren
#define IEX_EXPORT_CRYPTED   (1 << 10)
// Dokument/Attachment-Versionen
#define IEX_EXPORT_DOCVERS   (1 << 11)
#define IEX_EXPORT_ATTVERS   (1 << 12)
```

Anwendung:

```
(1 << 0) = 2^0  (Bit 0 gesetzt) 1
(1 << 1) = 2^1  (Bit 1 gesetzt) 2
(1 << 2) = 2^2  (Bit 2 gesetzt) 4
...
```

int Export(AnsiString sDestPath, int iExportType, int iParentId,
int iOptions, AnsiString sDocTypes, AnsiString sStartDate,
AnsiString sEndDate, AnsiString sObjList)

Parameter:

sDestPath	Zielpfad für die Exportdaten. Dieses Verzeichnis muß, wenn es vorhanden ist, leer sein.
iExportType	Reserviert, mit 0 zu füllen.
iParentId	1: Archiv ansonsten die ELO-ObjektID des Startknotens
iOptions	siehe Liste oben
sDocTypes	Leer: alle Dokumententypen, ansonsten ein Textstring mit 0 + 1 für jede Dokumententypnummer („10010111111111“ – Freie Eingabe (Typ 0) und Typ 3 und Typ 5..13) Achtung: Bitwerte werden von links gezählt und beginnen mit "0"
sStartDate	Datumseinschränkung – leer: keine Einschränkung
sEndDate	s.o.
sObjList	ELO ObjektIds der zu exportierenden Einträge (mit : getrennt, also z.B. „124:125:126“)

Rückgabe

-1	Kein Arbeitsbereich aktiv
-2	Fehler beim Export
-3	Zielpfad konnte nicht erstellt werden
1	Ok

Beachten Sie bitte, dass der Export in der Postbox des aktiven Anwenders eine Reportdatei schreibt, welche Sie zur Auswertung möglicher Fehler heranziehen sollten.

Funktion FindFirstWv

Diese Funktion kommt zur Anwendung, wenn Wiedervorlagetermine zu einem bestimmten ELO Objekt ermittelt werden müssen. Die Funktion erhält als Parameter die Objekt-ID des gewünschten ELO Objekts, sie liefert die Anzahl der gefundenen Wiedervorlagetermine zurück. Mit Hilfe der Funktion FindNextWv werden die Ids der Wiedervorlagetermine ausgelesen.

int FindFirstWv(int ObjektId)

Parameter:

ObjektId	ID des ELO Objekts

Rückgabewerte:

>=0: Anzahl der gefundenen Wiedervorlagetermine
-1: Kein Arbeitsbereich aktiv
-2: Fehler beim Lesen der Wiedervorlagetermine

siehe auch: FindNextWv

Funktion FindNextWv

Mit dieser Funktion werden die zu einem ELO Objekt gehörenden Wiedervorlageeinträge ausgelesen, zuvor muß FindFirstWv aufgerufen werden.

int FindNextWv()

Parameter:

Keine

Rückgabewerte:

>=0: ID des Wiedervorlagetermins
-1: keine Wiedervorlagetermin vorhanden

siehe auch: FindFirstWv

Funktion FindUser

Funktion FindUserEx

Diese Funktion sucht die interne ELO Anwendernummer zu einem Namen.

int FindUser(AnsiString UserName)
int FindUserEx(AnsiString UserName, int Mode)

Parameter:

UserName	Name des Anwenders zu dem die AnwenderId gesucht wird.
Mode	0: ELO Name, 1: NT Name, 2: Outlook Name.

Rückgabewerte:

AnwenderId oder –1 falls kein Anwender mit diesem Namen gefunden wurde

siehe auch: ActiveUserId
LoadUserName
SelectUser

Funktion FreezeDoc

Mit der Funktion FreezeDoc können Sie ein ELO Dokument über den Tiff-Printer in eine Tiff-Datei konvertieren und als neue Version an das logische Dokument anfügen.

Hierbei sind ein paar Rahmenbedingungen zu beachten:

- Der ELO Tiff-Printer muss in den Optionen angemeldet sein
- Das DruckTemp Verzeichnis muss vor der Befehlsausführung leer sein, das kann z.B. über ein UpdatePostbox sichergestellt werden.
- Während der Konvertierung dürfen keine konkurrierenden Druckbefehle aktiviert werden.
- Die Konvertierung findet über ein ShellExecute("print" ...) Aufruf statt. Deshalb muss auf dem Clientcomputer die entsprechende Applikation für das Dokument installiert sein. Zudem bringen manche Applikationen (je nach Dokumententyp/Inhalt) eigene Druckerdialoge auf.

Bekannte Probleme beim Drucken:

- (Outlook): Solange Outlook aktiv ist, reagiert es nicht auf Änderungen des Standarddruckers, die Umschaltung auf den Tiff-Printer wird also ignoriert. Beenden Sie deshalb vor der Konvertierung von MSG Dateien alle offenen Outlook-Fenster. Alternativ hierzu können Sie den Standarddrucker in Ihrem System fest auf den Tiff-Printer einrichten.
- (PowerPoint): Beim Konvertieren von PPT Dateien wird ein eigener Druckdialog angezeigt, der vom Anwender manuell quittiert werden muss.
- Der Ausdruck ist nur unter Windows NT, Windows 2000 und Windows XP möglich, da es unter Windows 95/98/ME zu viele Funktionseinschränkungen gibt.

Int FreezeDoc(int ObjectId)

Parameter:

ObjectId	logische Nummer des zu konvertierenden Dokuments

Rückgabewerte:

-1:	kein Arbeitsbereich aktiv
-2:	Fehler beim Ausdrucken
1:	ok

Beispiel:

```
Set Elo=CreateObject( "ELO.office" )

if Elo.SelectView(0)<>1 then
  MsgBox "Dieses Skript kann nur in der Archivansicht ausgeführt werden."
else
  ' über alle Dokumente des aktuellen Registers laufen
  for i=0 to 10000
    id=Elo.GetEntryId(i)
    if id<1 then exit for

    ' prüfen, ob es sich um ein Dokument handelt
    res=Elo.PrepareObjectEx( id,0,0 )
    if res>0 then
      if Elo.ObjType=Elo.ArchiveDepth then
        call Elo.FreezeDoc(id)
      end if
    end if
  next
end if
```

siehe auch:

Funktion FromClipboard

Diese Funktion übernimmt einen Text vom Windows Clipboard

AnsiString FromClipboard()

Parameter:

keine

Rückgabewerte:

Text aus dem Windows Clipboard

Siehe auch: ToClipboard

Funktion GetArcName

Diese Funktion ermittelt den aktuellen Archivnamen.

AnsiString GetArcName( )

Parameter:

keine

Rückgabewerte:

Archivname oder Leerstring bei Fehler

siehe auch:

Funktion GetArchiveName

Mittels der Funktion GetArchiveName können Sie den Archivnamen zu einer gegebenen Archivnummer ermitteln. Über die Archivnummer –1 erhalten Sie eine Liste aller verfügbaren Archive, getrennt durch das Systemtrennzeichen (normal ¶).

AnsiString GetArchiveName(int ArchiveNo)

Parameter:

ArchiveNo:	-1: Eine Liste aller verfügbaren Archive 0..n: Name zur gegebenen Archivnummer , leer wenn ungültig

Rückgabewerte:

ArchivName oder leer bei Fehler

Funktion GetAutoDlgValue (int Index)

Liefert den eingetragenen oder ausgewählten Wert der Dialogobjekte.

AnsiString GetAutoDlgValue (int Index)

Index : Position des Objektes beginnen mit 1 für das erste Objekt.

Rückgabewert :

Bei CheckBoxen und Radiobuttons 0 – Unchecked 1 – Checked
Bei Edit Feldern den Inhalt des Eingabefeldes.
Bei Labels wird ein Leerstring zurückgeliefert.

Bsp.:

Liefert den Text des Eingabe Feldes und Checked/Unchecked der Checkbox.

```
Elo.CreateAutoDlg (“Personen info“)
Elo.AddAutoDlgControl (4,1,“Name“,““)
Elo.AddAutoDlgControl (2,3,”Verheiratet? Ja”,”0”)
Elo.ShowAutoDlg
MsgBox GetAutoDlgValue (1)
MsgBox GetAutoDlgValue (2)
```

siehe auch: CreateAutoDlg
AddAutoDlgControl
ShowAutoDlg

Funktion GetCookie

Die Funktion GetCookie liest den Wert eines Cookie-Eintrags und gibt ihn an das aufrufende Programm zurück. Dieser Zugriff ist primär dazu gedacht, daß ELO Scripting Macros dauerhaft Informationen in ELO hinterlegen können. Der Cookie-Speicher wird beim Beenden von ELO gelöscht.

Der Zugriff auf ein Cookie erfolgt über den Namen (Ident), zurückgegeben wird der zugeordnete Wert. Falls das Cookie unbekannt ist wird ein Leerstring zurückgegeben.

Beachten Sie bitte, daß die Anzahl der Cookies begrenzt ist, jedes Makro oder jedes andere externe Programm sollte nur wenige davon verwenden und bei jedem Aufruf die gleichen wieder verwenden statt immer wieder neue anzulegen.

AnsiString GetCookie(AnsiString Ident)

Parameter:

Ident Cookie-Bezeichnung, wurde bei SetCookie vorgegeben

Rückgabewerte:

"": Unbekanntes oder leeres Cookie
ansonsten: Inhalt des angeforderten Cookies.

Siehe auch: SetCookie

Funktion GetDocExt

Gibt zu einer Objekt- oder DokumentenId die Externsion (Windows Dokumententyp) zurück.

AnsiString GetDocExt(int ObjDocId, int Status)

Parameter:

ObjDocId	Interne ELO Objekt- oder DokumentenId
Status	Bit 0: 0: DokumentenId, 1: ObjektId

Rückgabewerte:

Extension des Dokuments.

Beispiel:

```
SET Elo=CreateObject( "ELO.office" )
ID=Elo.GetEntryId(-1)
if ID>1 then
  call Elo.PrepareObjectEx( ID,0,0 )
  if Elo.ObjTypeEx=254 then
    Ext=UCase(Elo.GetDocExt( ID, 1 ))
    MsgBox "Dokumententyp zum Eintrag '" & Elo.ObjShort & "' : " & Ext
  end if
end if
```

siehe auch: UpdateDocument
InsertDocAttachment
GetDocumentPathVersion

Funktion GetDocFromObj

DieseFunktion ermittelt zu einer logischen Objekt-Id die dazu gehörende Dokumentenmanager Dokumenten-Id. Diese kann dann für dokumentenbezogene Aktionen (z.B. GetDocumentPathName oder GetDocumentSize) verwendet werden.

Über den Flags Parameter kann bestimmt werden, ob die aktuelle Arbeitsversion des Dokuments oder der Dateianbindung zurückgeliefert wird:

0	Aktuelle Dokumentenversion (Arbeitsversion) liefern
1	Aktuelles Attachment liefern

int GetDocFromObj(int ObjId, int Flags)

Parameter:

ObjId: Logische Objekt-Id des zu suchenden Eintrags
Flags: 0 oder 1

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: ObjId nicht gefunden
-3: Ungültiges Objekt
0: Keine Dokumentendatei zugeordnet
>1: Ok, Dokumenten-Id

Beispiel

```
Set Elo = CreateObject( "ELO.office" )
Id = Elo.GetEntryId(-1)
if Id > 1 then
  DId = Elo.GetDocFromObj( Id, 0 )
  if DId > 0 then
    MsgBox Elo.GetDocumentPathName( DId )
  end if
end if
```

Funktion GetDocRefComment

Über diese Funktion können Sie den Versionseintrag und Kommentar zu einer Dokumentenversion abfragen. Die beiden Felder werden in einem String, getrennt durch das Pipe-Symbol "|" zurückgegeben. Beachten Sie bitte, dass im Augenblick nur zwei Felder zurückgegeben werden, das kann sich in Zukunft ändern. Sie müssen in Ihren Skripten Rücksicht darauf nehmen, dass hier beliebig viele weitere Felder hinzukommen können.

AnsiString GetDocRefComment(int ObjectId, int DocumentId)

Parameter:

ObjectId	ELO Objektid des Dokuments
DocumentId	Accessmanager DokumentenId der Dokumentenversion

Rückgabewerte:

Kommentart+Versionstext, leer bei Fehler.

Beispiel:

```
set Elo=CreateObject( "ELO.office" )

ObjId=Elo.GetEntryId(-1)
if ObjId>0 then
  DocId=Elo.GetDocumentPathVersion( ObjId, 10, 0 )
  if DocId=0 then
    MsgBox "Es handelt sich nicht um ein Dokument oder" & vbcrlf & "das
Dokument besitzt kein Attachment"
  else
    DocInfos=Split(Elo.GetDocRefComment( ObjId + 1073741824, DocId ),"|")
    MsgBox DocInfos(1)
  end if
else
  MsgBox "Es ist kein Dokument aktiv"
end if
```

siehe auch: GetDocumentPathVersion

Funktion GetDocumentExt

Gibt die Dateikennung (Extension) zu einer Dokumentendateinummer zurück.Als DocId muss die Dokumentenmanager-Id der Datei und nicht die logische Objekt-Id angegeben werden. Sie erhalten diese Nummer z.B. über den Aufruf GetDocFromObj.

AnsiString GetDocumentExt(int DocId)

Parameter:

DocId	Dokumentenmanager Datei-Id

Rückgabewerte:

Extension oder Leer im Fehlerfall

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )
Id = Elo.GetEntryId(-1)
if Id > 1 then
  DId = Elo.GetDocFromObj( Id, 0 )
  if DId > 0 then
    MsgBox Elo.GetDocumentExt( DId )
  end if
end if
```

Funktion GetDocumentOrientation

Mit dieser Funktion können Sie eine Bilddatei in der Postbox analysieren lassen, um eine eventuelle Rotation um 90, 180 oder 270 Grad zu erkennen. Die Analyse erfolgt unter Verwendung des OCR-Systems. Vor Verwendung der Funktion muss die Funktion OCRInit aufgerufen werden, nach Beendigung die Funktion OCRExit. Mit Hilfe der Funktion RotateFile kann die Datei nach der Analyse so gedreht werden, dass Texte von unten lesbar sind (nur Singlepage TIFF-Dateien).

Als Dateiname kann ein Eintrag aus der Postbox übergeben werden (ohne Pfad, nur der Dateiname) oder ein Index in die Postliste (durch ein # gekennzeichnet, z.B. #0 ist der erste Eintrag in der Postliste).

int GetDocumentOrientation(AnsiString FileName, int PageNumber)

Parameter:

FileName:	Name der zu untersuchenden Datei
PageNumber:	Seitennummer (Seite 1 = „0“)

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehler beim Laden der Datei
-3: Fehler bei der Ermittlung der Rotation
1..4: Orientierung (1=0°, 2=90°, 3=180°, 4=270°)

Siehe auch: OrientFile
RotateFile
UpdatePostbox

Funktion GetDocumentPath

Liest aus einem Dokument die Dokument oder Attachment Datei und gibt einen Zugriffspfad auf die Datei zurück. Normale Archivdokumente sind schreibgeschützt, der Anwender kann deshalb über den Status eine freie Kopie anfordern (AUTO_WRITEACCESS). Diese Kopie ist allerdings nur begrenzte Zeit gültig und wird beim beenden von ELO automatisch gelöscht.

AnsiString GetDocumentPath(int DocId, int Status)

Parameter:

DocId	Interne ELO ObjektId
Status	Bit 0: Schreibzugriff auf Kopie Bit 1: Attachment statt Dokumentdatei Bit 2: Volltextrepräsentation statt Dokumentendatei Bit 3: Liste der Dokumentenids statt Dateipfad Bit 4: Signaturdatei statt Dokumentendatei Die Bits 1,2,3 und 4 schließen sich gegenseitig aus, sie dürfen nicht kombiniert werden.

Rückgabewerte:

Zugriffspfad auf das Dokument bzw. die Dateianbindung

siehe auch: UpdateDocument
InsertDocAttachment
GetDocumentPathVersion

Funktion GetDocumentPathName

Gibt den Dateipfad zu einer Dokumentendateinummer zurück. Dieser Dateipfad wird aus der Sicht des Dokumentenmanagers gebildet und ist im Allgemeinen vom Client aus nicht zugreifbar. Als DocId muss die Dokumentenmanager-Id der Datei und nicht die logische Objekt-Id angegeben werden. Sie erhalten diese Nummer z.B. über den Aufruf GetDocFromObj.

AnsiString GetDocumentPathName(int DocId)

Parameter:

DocId	Dokumentenmanager Datei-Id

Rückgabewerte:

Zugriffspfad auf das Dokument bzw. die Dateianbindung oder Leer im Fehlerfall

Verfügbar seit:

5.00.018

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )
Id = Elo.GetEntryId(-1)
if Id > 1 then
  DId = Elo.GetDocFromObj( Id, 0 )
  if DId > 0 then
    MsgBox Elo.GetDocumentPathName( DId )
 end if
end if
```

Funktion GetDocumentPathVersion

Liest aus einem Dokument die Dokument oder Attachment Datei und gibt einen Zugriffspfad auf die Datei zurück. Bei versionskontrollierten Dokumenten kann über den Parameter *Version* auf vorherige Versionen zugegriffen werden. Dabei wird *Version* mit 0 beginnend (=aktuelle Version) so lange um 1 erhöht, bis ein leerer String zurückgeliefert wird.
Normale Archivdokumente sind schreibgeschützt, der Anwender kann deshalb über den Status eine freie Kopie anfordern (AUTO_WRITEACCESS). Diese Kopie ist allerdings nur begrenzte Zeit gültig und wird beim beenden von ELO automatisch gelöscht.

AnsiString GetDocumentPathVersion(int DocId, int Status,int Version)

Parameter:

DocId	Interne ELO ObjektId
Status	Bit 0: Schreibzugriff auf Kopie Bit 1: Attachment statt Dokumentdatei Bit 2: Volltextreprästentation statt Dokument Bit 3: Dokumentennummern statt Dokument
Version	0 = aktuelle Version 1... = Versionsstände Falls Bit3 aus dem Status gesetzt ist, liefert 0 nur die aktuelle Nummer, 1 liefert alle Nummern

Rückgabewerte:

Zugriffspfad auf das Dokument bzw. die Dateianbindung
Falls Bit 3 aus dem Status gesetzt ist, enthält der Rückgabewert einen oder mehrere Zahlen, getrennt durch ein Pipe Symbol "|".

Beispiel:

```
ObjId=Elo.GetEntryId(-1)
if ObjId>0 then
  DocIds=Elo.GetDocumentPathVersion( ObjId, 10, 1 )
  if DocIds="" then
    MsgBox "Es handelt sich nicht um ein Dokument oder" & vbcrlf & "das
Dokument besitzt kein Attachment"
  else
    MsgBox DocIds
  end if
else
  MsgBox "Es ist kein Dokument aktiv"
end if
```

siehe auch: UpdateDocument
InsertDocAttachment
GetDocumentPath

Funktion GetDocumentSize

Gibt die Dateigröße zu einer Dokumentendateinummer zurück.Als DocId muss die Dokumentenmanager-Id der Datei und nicht die logische Objekt-Id angegeben werden. Sie erhalten diese Nummer z.B. über den Aufruf GetDocFromObj.

Int GetDocumentSize(int DocId)

Parameter:

DocId	Dokumentenmanager Datei-Id

Rückgabewerte:

-1 : Kein Arbeitsbereich aktiv
-2 : Fehler beim Lesen der Dokumentenmanager Information
>=0: Dateigröße.

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )
Id = Elo.GetEntryId(-1)
if Id > 1 then
  DId = Elo.GetDocFromObj( Id, 0 )
  if DId > 0 then
    MsgBox Elo.GetDocumentSize( DId )
  end if
end if
```

Funktion GetEntryId

Diese Funktion liefert die interne ELO ObjektId einer Zeile der Archiv-, Postbox- oder Suchansicht zurück. Über diesen Weg können Sie eine Liste aller (links) sichtbaren Objekte erhalten indem Sie beginnend mit Zeile 0 aufsteigend solange die Funktion GetEntryId aufrufen, bis Sie eine 0 zurückerhalten.
Wenn Sie den Aufruf in der Wiedervorlage starten, erhalten Sie statt der ObjektId eine WiedervorlageId zurück, sofern die Zeile einen Wiedervorlage-Eintrag darstellt. Handelt es sich bei der angesprochenen Zeile um einen Workflow-Eintrag, erhalten Sie die Objekt-ID des im Workflow befindlichen ELO Archivelements. Um Workflows und Wiedervorlagen voneinander zu unterscheiden, kann nach dem Aufruf von GetEntryID mit Hilfe des Properties IsWFLine geprüft werden.

Beim Aufruf der Funktion gibt es noch einige „spezielle“ Zeilen:

-1: Es wird der aktuell ausgewählte Eintrag gelesen
-2: Es wird die Nummer des internen EloObjekts gelesen

-10: Es wird die selektierte Zeile aus der Schrankliste gelesen
-11: Es wird die selektierte Zeile aus der Ordnerliste gelesen
-12: Es wird die selektierte Zeile aus der Registerliste gelesen
-13: Es wird die selektierte Zeile aus der Dokumentenliste gelesen

int GetEntryId(int Line)

Parameter

Line Zu lesende Zeile (0...) oder -1 ... -13

Rückgabewerte:

Interne ELO ObjektId
0: (Fehler, kein weiterer Eintrag, keine Auswahl vorgenommen)
-1: Kein Arbeitsbereich aktiv

siehe auch: GetEntryName
IsWFLine
ReadWFNode

Funktion GetEntryName

Über diese Funktion können Sie schnell die Kurzbezeichnung eines ELO Objektes über die interne ELO Objektnummer ermitteln. Falls Sie als ObjektId den Wert 0 einsetzen, erhalten Sie die Kurzbezeichnung des aktuell ausgewählten Eintrags.

AnsiString GetEntryName(int ObjId)

Parameter:

ObjId Elo-ObjektId oder 0 für den aktuell ausgewählten Eintrag

Rückgabewerte:

Kurzbezeichnung oder ein Leerstring im Fehlerfalle

siehe auch: GetEntryId

Funktion GetGuidFromObj

Diese Funktion ermittelt die ELO-Guid zu einer gegebenen ELO-ObjektId.

AnsiString GetGuidFromObj(long ObjId)

Parameter:

Guid	Guid des zu suchenden Objekts

Rückgabewerte

Leer:	kein Arbeitsbereich aktiv oder Objekt nicht gefunden
Sonst:	Guid

siehe auch: ObjGuid
GetObjFromGuid

Funktion GetHistDoc

Gibt die interne Dokumenten-Id zum n-ten Treffer aus LookupHistMD5 zurück.

int GetHistDoc(int TrefferNummer)

Parameter

TrefferNummer: 0..n (n wird von LookupHistMD5 zurückgegeben)

Rückgabewerte:

-1: Fehlerhafte TrefferNummer
>0: Dokumenten-Id

siehe auch: LookupHistMD5
GetHistObj
GetMD5Hash

Funktion GetHistObj

Gibt die interne ELO-Objekt-Id zum n-ten Treffer aus LookupHistMD5 zurück.

int GetHistObj(int TrefferNummer)

Parameter

TrefferNummer: 0..n (n wird von LookupHistMD5 zurückgegeben)

Rückgabewerte:

-1: Fehlerhafte TrefferNummer
>0: Objekt-Id

siehe auch: LookupHistMD5
GetMD5Hash
GetHistDoc

Funktion GetIndexGroups

Mittels der Funktion GetIndexGroups können Sie eine Liste aller eingetragenen Werte zu einer Indexzeile ermitteln, Duplikate werden dabei ignoriert. Wenn Sie z.B. eine Indexzeile "Artikelname" haben, dann können Sie über GetIndexGroups eine Liste aller vorhandenen Artikelnamen erzeugen. Diese können Sie dann zur Auswahl in einer Combo-Box verwenden. Beachten Sie bitte, dass diese Funktion nur sinnvoll eingesetzt werden kann, wenn die Treffermenge einige Dutzend bis einige Hundert Einträge nicht überschreitet.

Die Funktion kann in zwei unterschiedlichen Ausprägungen verwendet werden. Bei der direkten Liste (Artikel-Beispiel) wird nur eine Indexzeile beachtet. Sie können aber auch eine "indirekte" Liste erzeugen, z.B. alle Artikelnamen, die von dem Kunden mit der Kundennummer (KDNR) 4711 erworben worden sind. Hier sind zwei Indexzeilen betroffen, erst mal eine, welche die Selektion bestimmt und eine andere, die die aufzulistenden Elemente enthält. Hier ist aber noch stärker zu beachten, dass bei großen Treffermengen eine erhebliche Datenbanklast erzeugt wird.

AnsiString GetIndexGroups(AnsiString GroupCol, AnsiString SelCol, AnsiString SelVal)

Parameter:

GroupCol:	Gruppenname der Indexzeile aus der die Liste erzeugt werden soll
SelCol:	Gruppenname der Indexzeile aus der die Selektion verwendet werden soll
SelVal:	Indexeinschränkung

Rückgabewerte:

Trefferliste oder leer bei Fehler

Beispiel: Es wird eine Liste aller unterschiedlichen Einträge aus der Indexzeile ELOFAQBER zusammengestellt, die in der Indexzeile ELOVER die Zeichenfolge P3. enthalten.

```
set Elo=CreateObject("ELO.office")
Liste=Elo.GetIndexGroups( "ELOFAQBER", "ELOVER", "%P3.%" )
Liste=Replace( Liste, "¶", vbcrlf )
MsgBox Liste
```

Funktion GetLastDocId

Diese Funktion ermittelt die letzte physikalische Dokumentennummer im Archiv.

Int GetLastDocId()

Parameter:

keine.

Rückgabewerte:

-1: kein Arbeitsbereich aktiv
-2: Dokumentennummer konnte nicht ermittelt werden.
>0: Letzte Dokumenten-Id

siehe auch: Backup

Funktion GetLastVersionTimeStamp(int, int)

Gibt das Ablagedatum der pyhsikalischen Dokumentendatei zurück.

int GetLastVersionTimeStamp(int DocId, int Status)

Parameter

DocId : Elo ObjektId des Dokuments
Status: : 0: aktuelles Dokument, >1 Version
: 0x1000 als zusätzliches Flag dazu addiert: Attachment statt Dokument

Rückgabewerte:

-1: kein Arbeitsbereich aktiv
-2: kein Dokument gefunden
-3: ungültiger Status
-4: Dokumentendaten konnten nicht aus der Datenbank gelesen werden

>0: Datum im ELO internen Format

siehe auch: IntToDate

Funktion GetListEntry

Über diese Funktion können Sie eine Zeile aus der internen Trefferliste von CollectWv und DoInvisibleSearch auslesen. Das Ergebnis setzt sich aus folgenden Teilen zusammen:

Typ¶ObjId¶Owner¶Id1¶Id2¶Text

- Typ AC: Aktivität, OB: Objekt, WF: Workflowaufgabe, WV: Wiedervorlagetermin
- ObjId ELO-ObjektId
- Owner Eigentümer des Termins (immer 0 bei OB)
- Id1 OB: Attachement Id, WF: Workflow Id, sonst 0
- Id2 WF: Node Id, sonst 0

AnsiString GetListEntry(int Index)

Parameter:

Index Zeilennummer 0..(Anzahl der Treffer –1)

Rückgabewerte:

Leer: Kein Arbeitsbereich aktiv oder Zeile nicht vorhanden
Sonst: Zeileninhalt

Beispiel:

```
Set Elo=CreateObject("ELO.office")
UserId=Elo.ActiveUserId ' nur die eigenen Termine
'UserId=Elo.ActiveUserId + 1610612736 ' alle Termine

cnt=Elo.CollectWv( UserId, "22.06.2006", "30.06.2006", 3)
for i=0 to cnt-1
  msg=msg & Elo.GetListEntry(i) & vbcrlf
next
MsgBox msg
```

siehe auch: CollectWv
DoInvisibleSearch

Funktion GetMD5Hash

Über diese Funktion können Sie den MD5 Hash zu einer Datei oder zu einem Datenblock ermitteln. Wenn Sie als Parameter einen Dateinamen angeben, erhalten Sie den Hash zu der Datei. Wenn Sie einen String, beginnend mit den Zeichen „##“ übergeben, erhalten Sie den Hash Wert zu diesem String (die ##-Zeichen werden dabei nicht mitgezählt).

AnsiString GetMD5Hash(AnsiString FileName)

Parameter

FileName	Dateiname oder Eingabestring für den MD5 Hash Wert

Rückgabewert:

MD5 Hash Wert als 32 Zeichen HEX-String

siehe auch: LookupHistMD5
GetHistObj
GetHistDoc

Funktion GetObjAttrib

Diese Funktion liest den Wert einer Eingabezeile der aktuellen Dokumentenmaske aus.

Zu jeder Eingabezeile gehören 3 Werte:

Die „sichtbare“ Bezeichnung, sie wird in den Dokumentenmasken dem Anwender als Feldbezeichnung angezeigt (z.B. Rechunungsnummer). Sie dient als Benennung der Zeile für den Menschen.

Die Ergänzung, sie wird in der Datenbank zur Identifikation der Zeile im Index verwendet (z.B. RENR). Sie dient als Benennung der Zeile für die Maschine.

Der Eingabewert, hier wird die Anwendereingabe gespeichert (z.B. 199807106)

AnsiString GetObjAttrib(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes

Rückgabewerte:

Aktueller Inhalt des Eingabewertes

siehe auch: SetObjAttrib
GetObjAttribKey
GetObjAttribName
GetObjAttribFlags
GetObjAttribMin
GetObjAttribMax
GetObjAttribType

Funktion GetObjAttribFlags

Diese Funktion liest die Flags einer Eingabezeile der aktuellen Dokumentenmaske aus. Der Rückgabewert ist vom Typ Integer, die dort gesetzten Bits haben folgende Bedeutung:

Bit 0	Eintragungen nur mit Stichwortliste erlaubt
Bit 1	* automatisch vor Suchtext einfügen
Bit 2	* automatisch nach Suchtext einfügen
Bit 3	Neue Lasche nach dieser Zeile
Bit 4	Zeile unsichtbar
Bit 5	Zeile schreibgeschützt
Bit 6	Spalte mit hoher Priorität, Text in die Kurzbezeichnung aufnehmen

int GetObjAttribFlags(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes

Rückgabewerte:

Aktuelle Flags der angesprochenen Maskenzeile

siehe auch:
- GetObjAttrib
- GetObjAttribKey
- GetObjAttribName
- SetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

Funktion GetObjAttribKey

Diese Funktion liest die Indexbezeichnung einer Eingabezeile der aktuellen Dokumentenmaske aus.

Zu jeder Eingabezeile gehören 3 Werte:

Die „sichtbare“ Bezeichnung, sie wird in den Dokumentenmasken dem Anwender als Feldbezeichnung angezeigt (z.B. Rechunungsnummer). Sie dient als Benennung der Zeile für den Menschen.

Die Ergänzung, sie wird in der Datenbank zur Identifikation der Zeile im Index verwendet (z.B. RENR). Sie dient als Benennung der Zeile für die Maschine.

Der Eingabewert, hier wird die Anwendereingabe gespeichert (z.B. 199807106)

AnsiString GetObjAttribKey(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthällt den Dateinamen des Dokumentes

Rückgabewerte:

Aktueller Inhalt des Index-Bezeichnungsfeldes

siehe auch:
- GetObjAttrib
- SetObjAttribKey
- GetObjAttribName
- GetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

Funktion GetObjAttribMax

Diese Funktion liest die maximale Eingabelänge einer Eingabezeile der aktuellen Dokumentenmaske aus. Dieses Feld wird in der Standard-Eingabemaske innerhalb von ELO ausgewertet. Falls Sie die Eingaben über OLE Automation programmgesteuert vornehmen, müssen Sie selber auf die Einhaltung der minimalen und maximalen Längen einer Eingabe achten.

int GetObjAttribMax(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthällt den Dateinamen des Dokumentes

Rückgabewerte:

Maximale Länge der Eingabe, 0: keine Kontrolle, beliebige Länge

siehe auch: GetObjAttrib
GetObjAttribKey
GetObjAttribName
GetObjAttribFlags
GetObjAttribMin
SetObjAttribMax
GetObjAttribType

Funktion GetObjAttribMin

Diese Funktion liest die minimale Eingabelänge einer Eingabezeile der aktuellen Dokumentenmaske aus. Dieses Feld wird in der Standard-Eingabemaske innerhalb von ELO ausgewertet. Falls Sie die Eingaben über OLE Automation programmgesteuert vornehmen, müssen Sie selber auf die Einhaltung der minimalen und maximalen Längen einer Eingabe achten.

int GetObjAttribMin(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthällt den Dateinamen des Dokumentes

Rückgabewerte:

Minimale Länge der Eingabe, 0: keine Kontrolle, beliebige Länge

siehe auch: GetObjAttrib
GetObjAttribKey
GetObjAttribName
GetObjAttribFlags
SetObjAttribMin
GetObjAttribMax
GetObjAttribType

Funktion GetObjAttribName

Diese Funktion liest die Bezeichnung einer Eingabezeile der aktuellen Dokumentenmaske aus.

Zu jeder Eingabezeile gehören 3 Werte:

Die „sichtbare“ Bezeichnung, sie wird in den Dokumentenmasken dem Anwender als Feldbezeichnung angezeigt (z.B. Rechunungsnummer). Sie dient als Benennung der Zeile für den Menschen.

Die Ergänzung, sie wird in der Datenbank zur Identifikation der Zeile im Index verwendet (z.B. RENR). Sie dient als Benennung der Zeile für die Maschine.

Der Eingabewert, hier wird die Anwendereingabe gespeichert (z.B. 199807106)

AnsiString GetObjAttribName(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthällt den Dateinamen des Dokumentes

Rückgabewerte:

Aktueller Inhalt des Bezeichnungs-Feldes

siehe auch:
GetObjAttrib
GetObjAttribKey
SetObjAttribName
GetObjAttribFlags
GetObjAttribMin
GetObjAttribMax
GetObjAttribType

Funktion GetObjAttribType

Diese Funktion liest die Art der Eingabe einer Eingabezeile der aktuellen Dokumentenmaske. Dieses Feld wird in der Standard-Eingabemaske innerhalb von ELO ausgewertet. Falls Sie die Eingaben über OLE Automation programmgesteuert vornehmen, müssen Sie selber auf die Einhaltung der Art einer Eingabe achten.

0: beliebiges Textfeld
1: Datumsfeld
2: Numerisches Feld
3: ISO-Datum
4: Listeneintrag
5: Anwenderfeld
6: Thesaurus
7: Numerisch mit fester Breite
8: Numerisch mit fester Breite, 1 Nachkommastelle
9: Numerisch mit fester Breite, 2 Nachkommastelle
10: Numerisch mit fester Breite, 4 Nachkommastelle
11: Numerisch mit fester Breite, 6 Nachkommastelle
12: Aktenzeichen

int GetObjAttribType(int AttribNo)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthällt den Dateinamen des Dokumentes

Rückgabewerte:

Art der zulässigen Eingabe

siehe auch: GetObjAttrib
GetObjAttribKey
GetObjAttribName
GetObjAttribFlags
GetObjAttribMin
GetObjAttribMax
SetObjAttribType

Funktion GetObjFromDoc

Diese Funktion ermittelt zu einer Dokumenten-Id das zugehörende Objekt (die Dokumenten-Id gehört zu der Datei, nicht zu der Verschlagwortung – diese Id wird hier gerade ermittelt).

int GetObjFromDoc(int DocId)

Parameter:

DocId	Dateinummer des Quelldokuments

Rückgabewerte:

-1	kein Arbeitsbereich aktiv
0	DocId nicht gefunden, keine Verschlagwortung zugeordnet
>0	Objekte Id der Verschlagwortung zu der Datei

Funktion GetObjFromDocEx

DieseFunktion ermittelt zu einer AccessManager Document-Id das zugehörende Dokument. Im Gegensatz zur Funktion GetObjFromDoc wird hier nicht nur das aktuelle Arbeitsdokument sondern auch das Attachment sowie die Dokumenten- und Attachmentversionen berücksichtig. Die Entscheidung, welche Teile bei der Suche Beachtung finden sollen, wird über den Flags Parameter gesteuert:

2	Nur aktuelle Dokumentenversion (Arbeitsversion) berücksichtigen
3	Nur aktuelles Attachment berücksichtigen
4	Alle Dokumentenversionen berücksichtigen
5	Alle Attachmentversionen berücksichtigen
6	Alle Dokumenten- und Attachmentversionen berücksichtigen

int GetObjFromDocEx(int DocId, int Flags)

Parameter:

DocId: AccessManager Dokumentennummer des zu suchenden Eintrags
Flags: 0..4

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-3: DocId nicht gefunden
>1: Ok, Objekt-Id

Beispiel

```
function Check( DocId )
  s="AM-DocId: " & DocId & vbcrlf
  s=s & "Doc: " & Elo.GetObjFromDocEx( DocId, 0 ) & vbcrlf
  s=s & "Att: " & Elo.GetObjFromDocEx( DocId, 1 ) & vbcrlf
  s=s & "DocV: " & Elo.GetObjFromDocEx( DocId, 2 ) & vbcrlf
  s=s & "AttV: " & Elo.GetObjFromDocEx( DocId, 3 ) & vbcrlf
  s=s & "DAV: " & Elo.GetObjFromDocEx( DocId, 4 ) & vbcrlf &vbcrlf
  Check=s
end function


set elo=CreateObject("ELO.office")

v1=Check(1)
v2=Check(2)
v3=Check(3)
v4=Check(4)
v5=Check(5)

MsgBox v1 & v2 & v3 & v4 & v5
```

Funktion GetObjFromGuid

Diese Funktion ermittelt die ELO ObjektId zu einer gegebenen ELO-Guid.

int GetObjFromGuid(AnsiString Guid)

Parameter:

Guid	Guid des zu suchenden Objekts

Rückgabewerte

-1	kein Arbeitsbereich aktiv
0	Guid nicht gefunden
>0	Objekt-Id

siehe auch: GetGuidFromObj
ObjGuid

Funktion GetObjMaskNo

Gibt die Nummer der aktuellen Maske (Dokumententyp) zurück. Eine analoge SetObjMaskNo hierzu gibt es nicht, bei neuen Elo-Dokumenten wird der Dokumententyp einmal mit dem PrepareObject „für alle Zeiten“ festgelegt.

int GetObjMaskNo()

Parameter:

keine

Rückgabewerte:

Nummer der aktuellen Dokumentenmaske

siehe auch: ReadObjMask
WriteObjMask

Funktion GetObjRef (int ObjId, int RefNo)

ELO bietet neben der hierarchischen Baumstruktur (Schrank - Ordner - Register - Dokument) die Möglichkeit, daß Sie weitere Referenzen anlegen können. Ein Dokument „Rechnung Müller“ kann im Register „Rechnungen“ abgelegt werden und mit einer zusätzlichen Referenz im Register „Müller“ eingetragen werden. Obwohl es das Dokument dann nur einmal im System gibt, ist es von beiden Stellen aus sichtbar und bearbeitbar.
Mit dieser Funktion kann der Pfad der Referenz ermittelt werden.

AnsiString GetObjRef (int ObjId, int RefNo)

ObjId : Interne ELO Id
RefNo : Nummer der anzuzeigenden Referenz (beginnend mit 0)
(0 ist die eigene Referenz)

Rückgabewert:
-1 - Kein Arbeitsbereich aktiv
-2 - Referenz nicht vorhanden

Ansosnst Pfad zur Referenz getrennt mit dem Separator

Bsp.:
Gibt den Pfad der zweiten Referenz von Objekt mit der Id 245 zurück.
Ref2 = Elo.GetObjRef (245,2)

siehe auch: InsertRef
RemoveRef

Funktion GetPopupObjectId()

Über diese Funktion können Sie das aktuell für ein Kontextmenü ausgewählte Objekt ermitteln. Dieser Eintrag ist nur dann gültig, wenn er in Folge eines Kontextmenüereignisses aufgerufen worden ist.

int GetPopupObjectId( )

Parameter:

keine

Rückgabewerte:

-1: kein Arbeitsbereich aktiv
>0: aktuelle ObjectId

siehe auch:

Funktion GetPostDir

Über diese Funktion können Sie schnell den Postboxpfad des aktuellen Anwenders ermitteln.

AnsiString GetPostDir( )

Parameter:

keine

Rückgabewerte:

Postboxpfad oder ein Leerstring im Fehlerfalle

Funktion GetScriptButton

Mit dieser Funktion kann die Belegung der Skriptbuttons innerhalb des ELO Hauptbildschirms abgefragt werden.
Die Funktion wird innerhalb von Installationsskripten aufgerufen, mit deren Hilfe Skripte importiert werden können bei gleichzeitiger Einrichtung der Button- und Menübelegung.

AnsiString GetScriptButton(int iTabSheet, int iButton)

Parameter:

iTabSheet	Auswahl der Ansicht: 1: Archivansicht (bis zu 16 Buttons) 2: Klemmbrett (bis zu 8 Buttons) 3: Postbox (bis zu 8 Buttons) 4: Recherche (bis zu 8 Buttons) 5: Wiedervorlage (bis zu 8 Buttons)
iButton	Nummer des Buttons (1..16 bzw. 1..8)

Rückgabewerte:

Name des Skripts

Siehe auch: SetScriptMenu
SetScriptButton
ImportScript

Funktion GetScriptEvent (AnsiString Event, int Mode)

Diese Funktion liest den Scriptnamen bzw. kompletten Pfad des in den Script-Events eingestellten Scripts.

AnsiString GetScriptEvent (AnsiString Event, int Mode)

Event : String mit dem Event-Bezeichner (siehe Event-Liste unter SetScriptEvent)
Oder vorangestellt mit # die Position (diese Werte können in kommenden ELO Versionen variieren.

Mode : 0 – Scriptname mit komplettem Pfad und Dateiendung
1 – Nur der Scriptname

Rückgabewerte :

-2 – Unbekannter Event
-1 – Modus nicht implementiert
0 – Kein Script vorhanden

Bsp.:

Liefert, sofern belegt, den kompletten Pfad mit Dateinamen des Scripts
Pfadkomplett = ELO. GetScriptEvent (“sEonTimer“,0)

Liefert nur den Script Namen zugriff über Event-Bezeichner
NurName = ELO. GetScriptEvent (“sEonTimer“,1)

Liefert den Script Namen zugriff über Event-Position (#0 = “sEonTimer“)
NurName = ELO. GetScriptEvent (“#0”,1)

siehe auch: SetScriptEvent

Funktion GetTreePath(int Mode, AnsiString Delimiter, int MaxLength)

Diese Funktion gibt den Archivpfad zu der aktuellen TreeView Auswahl zurück.

AnsiString GetTreePath (int Mode, AnsiString Delimiter, int MaxLength)

Mode : 0: Kurzbezeichnung, 1: ObjectId

Delimiter: Treensymbol zwischen den einzelnen Einträgen

MaxLength: Maximale Länge der Ausgabe. Wenn diese überschritten wird, dann kürzt ELO einen Teil heraus.

Rückgabewerte :

-2 – Unbekannter Event
-1 – Modus nicht implementiert
0 – Kein Script vorhanden

Beispiel:

```
Set Elo = CreateObject( „ELO.office“)
MsgBox Elo.GetTreePath( 0, „ // „, 127 )
MsgBox Elo.GetTreePath( 1, „:“, 1000 )
```

Funktion GotoId

Über diese Funktion können Sie ELO veranlassen auf ein bestimmtes Dokument (oder Schrank, Ordner, Register) zu wechseln. Wenn Sie eigene Einträge mit ELO Dokumenten verknüpfen wollen, reicht es also aus, wenn Sie die zugehörige ELO ObjektId speichern und bei Bedarf die Anzeige dieses Eintrags über GotoId anfordern. Über den Sonderfall GotoId(1) können Sie mit einem Schritt an die anfängliche Archivansicht zurückkehren.

Im Normalfalle finden Sie die Archivansicht dann so, daß das gewünschte Objekt in der linken Liste erscheint und selektiert ist. Es werden dann die Untereinträge (oder das Image falls das Objekt ein Dokument ist) angezeigt. Manchmal möchte man aber nicht das Objekt sondern direkt den Inhalt des Objektes anzeigen. In diesem Fall muß die ObjektId mit einem negativen Vorzeichen versehen werden.

int GotoId(int ObjektId)

Parameter:

ObjektId	Anzuwählendes ELO Objekt oder 0 (zurück zum Archivanfang)

Rückgabewerte:

-4: Der Anwender besitzt kein Leserecht
-3: Pfad zum Objekt nicht gefunden
-2: Objekt nicht gefunden
-1: Kein Arbeitsbereich aktiv
1: ok

siehe auch: LookupIndex
GetEntryId
GetEntryName

Funktion Import

Über diese Funktion können Sie einen Exportdatensatz wieder importieren.

int Import(AnsiString sSourcePath, int iParentId, int iSelId)

Parameter

sSourcePath:	Quellpfad des Exportdatensatzes
iParentId:	Vorgängerknoten des Importziels (1=Archiv)
iSelId:	ELO ObjektId des Importziels

Rückgabe

-1	Kein Arbeitsbereich aktiv
-2	Fehler beim Import
2	Ok

Beachten Sie bitte, dass der Import in der Postbox des aktiven Anwenders eine Reportdatei schreibt, welche Sie zur Auswertung möglicher Fehler heranziehen sollten.

Funktion ImportScript

Mit dieser Funktion kann ein Skript (*.VBS-Datei) in das ELO Skript-Verzeichnis importiert werden.
Die Funktion wird innerhalb von Installationsskripten aufgerufen, mit deren Hilfe Skripte importiert werden können bei gleichzeitiger Einrichtung der Button- und Menübelegung.

int ImportScript(AnsiString SourceFile, int ForceOverwrite)

Parameter:

SourceFile	Pfad+Name der Skriptdatei (z.B. „A:\Skript1.VBS“)
ForceOverwrite	Überschreiben eines eventuell bereits vorhandenen Skripts erzwingen

Rückgabewerte:

-3: Quelldatei nicht vorhanden
-2: Fehler beim Kopieren
-1: kein aktiver Arbeitsbereich vorhanden
1: ok

Siehe auch: SetScriptMenu
SetScriptButton
GetScriptButton

Funktion InsertDocAttachment

Mithilfe dieser Funktion können Sie an ein bestehendes ELO Dokument eine Dateianbindung anfügen. Beachten Sie bitte, daß je nach eingestellten Dokumentoptionen eine evtl. bereits vorhandene Dateianbindung in die Historyliste verschoben oder überschrieben wird. Wenn das Dokument den Status „Revisionssicher“ besitzt, wird dieser Funktionsaufruf zurückgewiesen.

int InsertDocAttachment(int ParentDoc, AnsiString DocumentFile)

Parameter:

ParentDoc	Interne ELO ObjektId des Dokuments an das die Datei angebunden werden soll.
DocumentFile	Zugriffspfad und Name der Datei welche angebunden werden soll.

Rückgabewerte:

-2: Operation mit Archiv- oder Datenbankfehler abgebrochen
-1: kein aktiver Arbeitsbereich vorhanden
1: ok

siehe auch: UpdateDocument
GetDocumentPath

Funktion InsertDocAttachmentEx

Mithilfe dieser Funktion können Sie an ein bestehendes ELO Dokument eine Dateianbindung anfügen. Beachten Sie bitte, daß je nach eingestellten Dokumentoptionen eine evtl. bereits vorhandene Dateianbindung in die Historyliste verschoben oder überschrieben wird. Wenn das Dokument den Status „Revisionssicher“ besitzt, wird dieser Funktionsaufruf zurückgewiesen.

int InsertDocAttachmentEx(int ParentDoc, String DocumentFile, String Comment, String Version)

Parameter:

ParentDoc	Interne ELO ObjektId des Dokuments an das die Datei angebunden werden soll.
DocumentFile	Zugriffspfad und Name der Datei welche angebunden werden soll.
Comment	Kommentar (z.B. Originaldateiname) zum Attachment
Version	Interne, frei vergebbare Versionsnummer oder Bezeichnung

Rückgabewerte:

-2: Operation mit Archiv- oder Datenbankfehler abgebrochen
-1: kein aktiver Arbeitsbereich vorhanden
1: ok

siehe auch: UpdateDocument
GetDocumentPath

Funktion InsertRef

ELO bietet neben der hierarchischen Baumstruktur (Schrank - Ordner - Register - Dokument) die Möglichkeit, daß Sie weitere Referenzen anlegen können. Ein Dokument „Rechnung Müller“ kann im Register „Rechnungen“ abgelegt werden und mit einer zusätzlichen Referenz im Register „Müller“ eingetragen werden. Obwohl es das Dokument dann nur einmal im System gibt, ist es von beiden Stellen aus sichtbar und bearbeitbar.

Der Parameter OldParent bestimmt, ob Sie eine neue Referenz anlegen wollen (OldParent=-1) oder eine bestehende Referenz verschieben wollen (OldParent>1).

Falls Sie sicher sind, daß Ihre Parameter absolut korrekt sind, können Sie über den Parameter CheckTypes die Typenkontrolle aus Effizienzgründen abschalten. In diesem Falle sind Sie dafür verantwortlich, daß die Referenz Typengerecht ausgeführt werden, ein Register darf also nur aus einem Ordner heraus, nicht aber aus einem Schrank oder einem anderen Register heraus referenziert werden.

Falls es von "NewParent" nach "ObjId" bereits eine Verbindung gibt, wird das nicht als Fehler gemeldet. Es wird auch keine doppelte Verbindung angelegt, es bleibt bei der ursprünglichen Situation.

Mit Hilfe dieser Funktion kann man auch ein Objekt verschieben (OldParentID mit einer ObjektID setzen)

int InsertRef(int ObjId, int OldParent, int NewParent, int CheckTypes)

Parameter:

ObjId	Interne ELO ObjektId des Eintrags auf den die Referenz zeigt
OldParent	-1=neue Referenz, ansonsten wird eine bestehende Referenz verschoben
NewParent	ObjektId des Eintrags von dem die Referenz ausgeht
CheckTypes	0: keine Prüfung 1: Prüfung durchführen

Rückgabewerte:

-10: Datenbankfehler beim Eintragen der Referenz
-6: Ein Dokument kann keine Untereinträge erhalten
-5: Fehler beim Lesen der neuen Vorgängerdaten
-4: Fehler beim Lesen der alten Vorgängerdaten
-3: Fehler beim Lesen der Objektdaten
-2: Fehler beim Verschieben der Referenz in der Datenbank
-1: kein Arbeitsbereich aktiv

siehe auch: MoveToArchive
LookupIndex

Funktion IntToDate

Diese Funktion wandelt einen ELO-internen Datumswert in einen Datumstext um.

AnsiString IntToDate(int Datum)

Parameter:

Datum Zu wandelnder Datumswert.

Rückgabewerte:

Datumstext

siehe auch: DateToInt

Funktion LoadPostImg

Diese Funktion lädt eine Datei in den Viewer der aktuellen Postboxansicht. Das Dokument muss sich dabei nicht in der ELO Postbox befinden.

int LoadPostImg(AnsiString FileName, int Flags)

Parameter:

FileName	Zu ladende Datei, incl. Pfad
Flags	reserviert, mit 0 zu belegen

Rückgabewerte:

-1	kein Arbeitsbereich aktiv
-2	Fehler beim Laden
1	ok

Funktion LoadUserName

Diese Funktion ermittelt den Anwendernamen zu einer Anwender Id. Falls der Anwender nicht existiert oder eine ungültige Anwendernummer übergeben wurde, wir ein Leerstring zurückgeliefert.

AnsiString LoadUserName(int UserId)

Parameter:

UserId Interne ELO Anwendernummer.

Rückgabewerte:

Anwendername

siehe auch: ActiveUserId
FindUser
SelectUser

Funktion LockObject

Wenn Sie einen bestehenden Eintrag bearbeiten wollen und sicherstellen müssen, daß er nicht gleichzeitig von anderer Stelle verändert wird, können Sie ihn über diese Funktion zu Beginn der Arbeit sperren und nach Beendigung wieder freigeben. Falls während dieser Zeit ein anderer Mitarbeiter dieses Objekt bearbeiten möchte, bekommt er die Mitteilung, daß dieser Eintrag bereits gesperrt ist.

int LockObject(int ObjektId, int ActionCode)

Parameter:

ObjektId	Interne ELO ObjektId für das zu sperrende oder freizugebende Objekt
ActionCode	0=freigeben, 1=sperren, 3=sperre abfragen

Rückgabewerte:

-4: (bei ActionCode 3) Der Datensatz ist nicht gesperrt
-3: Ungültiger ActionCode
-2: Sperre fehlgeschlagen (wird z.B. gerade von einem anderen Anwender gehalten)
-1: Kein Arbeitsbereich aktiv
1: (ActionCode 0 oder 1) ok
0..n: (ActionCode 3) Anwendernummer des Eigentümer der Sperre.

Funktion Login

Falls Ihr Programm ein bereits laufendes ELO vorfindet, kann es im Kontext des bereits angemeldeten Anwenders arbeiten.

Falls Ihr Programm jedoch ein eigenes ELO starten muß (z.B. ein Serverprozess), ist es notwendig, daß Sie als erste Aktion ein Login durchführen. Im Rahmen dieses Login übergeben Sie den gewünschten Anmeldenamen, das Paßwort und das zu bearbeitende Archiv.

Nach Beendigung der Arbeit ist dann unbedingt ein Logout (Login mit Anmeldenamen „LOGOUT“) durchzuführen, andernfalls wird das System beim nächsten Start einen möglichen Zugriffskonflikt bemängeln. Zum Verlassen des Systems haben Sie zwei Optionen, ein nomales Beenden (mit Beenden des Programms) über den Loginnamen „LOGOUT“ und ein Teilbeenden in den Logindialog hinein (z.B. um eine Neuanmeldung mit einem anderen Namen oder ein anderes Archiv durchzuführen) über "LOGOUTNOQUIT".

Falls Sie ein komplettes Logout durchgeführt haben, dürfen Sie nicht sofort ein neues Login ausführen. Da sich das Programm dann gerade im Beenden Zustand befindet, erhalten Sie in diesem Fall eine Schutzverletzung. Sie müssen nach einem kompletten Logout eine kurze Zeit warten (1..5 Sekunden) bis das Programm völlig beendet wurde UND ein neues Elo-Objekt mit CreateOleObject erzeugen (das alte ist nach dem Logout nicht mehr gültig, eine Verwendung führt zu einer Schutzverletzung).

int Login(AnsiString UserName, AnsiString Password, AnsiString ArchiveName)

Parameter:

UserName	ELO Anmeldename unter dem die Aktionen durchgeführt werden sollen.
Password	ELO Zugangs-Passwort
ArchiveName	benötigtes Archiv

Rückgabewerte:

-1: Login gescheitert
1: Logout ok
0..255: Login ok, Rückgabe der Verbindungsnummer

Property LookupDelimiter (AnsiString)

Diese Funktion ermittelt oder setzt das Trennsymbol für die LookupIndex Funktion (und für verwandte Aktionen wie z.B. den COLD-Spaltenindex). Diese Änderung wirkt sich auf den gesamten ELO Betrieb aus, nicht nur auf Aktivitäten der OLE-Automation Schnittstelle.

siehe auch: LookupIndex

Funktion LookupDocType

Ermittelt zu einem Dateinamen anhand der Extension den voreingestellten ELO Dokumententyp. Der Dateiname kann vollständig mit Pfad vorliegen, nur als Dateiname oder auch nur als Extension.

int LookupDocType(AnsiString FileName, int DefaultType)

Parameter:

FileName	Dateiname mit Extension.
DefaultType	Rückgabewert für den Fall, dass keine Zuordnung gefunden wurde

Rückgabewerte:

Dokumententyp oder DefaultType

Beispiel:

```
Set Elo=CreateObject("ELO.office")

Dim Exts
Exts=Array("x.msg", ".msg", "msg", "c:\d\x.doc", "y.xls")

res="Ext: Type" & vbcrlf

for i=LBound(Exts) to UBound(Exts)
  res=res & vbcrlf & Exts(i) & " :  " & Elo.LookupDocType( Exts(i), -1 )
next

MsgBox res
```

siehe auch:

Funktion LookupIndex

Ermittelt die interne ELO ObjektId über einen Zugriffspfad. Hierzu übergeben Sie einen ObjektIndex auf den gesuchten Eintrag und Sie erhalten die zugehörende ObjektId zurück. Über diese Funktion können Sie z.B. die ObjektId für ein bestimmtes Register suchen, in das Sie ein neues Dokument ablegen wollen.

Dieser ObjektIndex kann verschiedene Formen aufweisen:

Eintrag in die Chaosablage (nur für Dokumente erlaubt):

Das Property ObjIndex enthält einen leeren Eintrag (ObjIndex=""). Es wird also kein Ablageort vorgegeben, das Dokument erscheint also nicht in der Aktenstruktur, es kann nur über die Recherche angezeigt werden.

Eintrag über eine interne ELO Objekt-ID:

ObjIndex enthält die ObjektId des Vorgängerknotens (ObjIndex="#12345"), angeführt von dem Symbol #. Es liegt in Ihrer Verantwortung sicherzustellen, daß dieser Vorgängerknoten existiert und vom richtigen Typ ist.

Eintrag über einen Zugriffspfad:

ObjIndex enthält einen Zugriffspfad auf den Vorgängerknoten, beginnend mit dem Symbol ¶ (ObjIndex="¶Schrank¶Ordner¶Register"). Dieser Pfad setzt sich aus den Kurzbezeichnungen, getrennt durch das Symbol ¶, („¶"=Alt 0182) zusammen. Achten Sie bitte darauf, daß bei dieser Vorgehensweise innerhalb von einer Ebene nicht zweimal der gleiche Begriff auftreten darf, da sonst keine eindeutige Zuordnung erfolgen kann. Im Ordner Rechnungen darf also nicht zweimal ein Register März auftreten. Allerdings kann das Register März in jedem beliebigen anderen Ordner verwendet werden.

Eintrag über einen Schlüsselbegriff (Nur für Dokumente):

Sie können in einem Register bis zu drei Schlüsselbegriffe hinterlegen (z.B. KDNR 123). Wenn Sie in ObjIndex dann den Text KDNR=123 hinterlegen, wird das oben genannte Register als Vorgängerknoten verwendet.

Eintrag über *Schlüsseltext (???)

int LookupIndex(AnsiString ObjektIndex)

Parameter:

ObjIndex Zugriffspfad auf das gesuchte Objekt

Rückgabewerte:

Gesuchte ObjektId
-2: Fehler
-1: Kein Arbeitsbereich aktiv

Funktion LookupKeyName

Ermittelt die interne ELO KeyId über den Schlüsselnamen. Beachten Sie bitte, daß diese Information aus einem Client-Lokalen Cache kommt und neu angelegte Schlüssel anderer Stationen evtl. erst nach einem Neustart verfügbar sind.

int LookupKeyName(AnsiString KeyName)

Parameter:

KeyName	Name des Schlüssels dessen Nummer ermittelt werden soll

Rückgabewerte:

>=0: Key ID
-1: Schlüssel nicht gefunden

siehe auch: ObjKey
LookupUserName

Funktion LookupMaskName

Ermittelt die interne ELO MaskId über den Maskennamen. Beachten Sie bitte, daß diese Information aus einem Client-Lokalen Cache kommt und neu angelegte Masken anderer Stationen evtl. erst nach einem Neustart verfügbar sind.

int LookupMaskName(AnsiString MaskName)

Parameter:

MaskName	Name der Maske dessen Nummer ermittelt werden soll

Rückgabewerte:

>=0: Masken ID
-1: Maske nicht gefunden
-2: Kein Maskenname angegeben
-3: Kein Arbeitsbereich aktiv

siehe auch: ReadObjMask
WriteObjMask

Funktion LookupUserName

Ermittelt die interne ELO User- oder GroupId über den Anwender- oder Gruppennamen. Beachten Sie bitte, daß diese Information aus einem Client-Lokalen Cache kommt und neu angelegte Anwender anderer Stationen evtl. erst nach einer Zeitverzögerung oder einem Neustart verfügbar sind.

int LookupUserName(AnsiString KeyName)

Parameter:

UserName	Name des Anwenders/Gruppe dessen Nummer ermittelt werden soll

Rückgabewerte:

>=0: User/Group ID
-1: Name nicht gefunden

Verfügbar seit:

4.00.054

Beispiel:

```
Set Elo=CreateObject("ELO.office")
Name=InputBox("Bitte einen Anwender- oder Gruppennamen eingeben")
MsgBox Elo.LookupUserName(Name)
```

siehe auch: LookupKeyName
ReadUser

Property MaskFlags (int)

Das Property MaskFlags enthält Informationen über die Art der Maske und Vorgaben der Dokumenten-Flags für neu erzeugte Dokumente dieses Typs.

Bit 0,1:	00 keine Versionskontrolle, Dokument kann beliebig verändert werden. 01 Versionskontrollierte Ablage, alte Versionen bleiben erhalten. 10 Dokumentenechte Ablage, keine Bearbeitung mehr möglich. 11 reserviert
Bit 3:	0 nicht als Recherchemaske verwendbar 1 als Recherchemaske verwendbar
Bit 4:	0 nicht als Ablagemaske verwendbar 1 als Ablagemaske verwendbar
Bit 6:	0 keine Volltextnachbearbeitung. 1 zur Volltextnachbearbeitung anmelden.

Alle anderen Bits sind reserviert oder werden für interne Zwecke benötigt, sie sind mit 0 zu belegen.

siehe auch: ObjFlags

Property MaskKey (int)

Das Property MaskKey bestimmt den Schlüssel einer Maskendefinition. Die Schlüsselnummer ergibt sich aus der Tabelle "Schlüsselverwaltung". In ELO "Systemverwaltung -> Schlüssel...".

siehe auch: DocKey
DocKey
DocKind
DocPath

Property MaxResultSet(int)

Dieses Property enthält den Optionen-Eintrag für die maximale Größe der Trefferliste bei einer Suche. Beachten Sie bitte, dass dieser Wert nicht immer Anwendung findet. An einigen Stellen wird intern ein größerer Wert oder ein fester Wert eingesetzt.

Falls Sie diesen Wert in einem Script für eine eigene Suche verändern, sollten Sie ihn nach der Suche auf den originalen Wert zurücksetzen. Andernfalls verändern Sie diesen Wert auch für anschließende manuelle Suchen.

Verfügbar seit 5.00.180

Beispiel:

```
MsgBox "Aktueller Wert: " & Elo.MaxResultSet
Elo.MaxResultSet = 5000
```

Funktion MergeImages

Funktion MergeImagesEx

Diese Funktion ermöglicht es Ihnen, in eine Tiff Datei eine andere Tiff Datei einzukopieren (z.B. zum Schwärzen von Bildbereichen). Hierzu geben Sie die Quelldatei, die Zieldatei, die jeweiligen Seiten, die Transparenzfarbe und die Position im Zieldokument an. Beachten Sie bitte, dass die Transparenzfarbe als 24 Bit RGB Wert angegeben werden muss, auch bei reinen Schwarz/Weiss Dokumenten.
Als Basis für die Merge-Operation können 1 Bit (Schwarz/Weiss) und 24 Bit (Farbbilder) verwendet werden, Quelle und Ziel müssen die gleiche Farbauflösung besitzen. 4, 8 oder 16 Bit Zwischenformate werden nicht unterstützt.

int __fastcall EloServer::MergeImages (AnsiString sFileSrc, AnsiString sFileTgt, int iPageSrc, int iPageTgt, int lTransp, int X, int Y)
int __fastcall EloServer::MergeImagesEx (AnsiString sFileSrc, AnsiString sFileTgt, int iPageSrc, int iPageTgt, int lTransp, int X, int Y, int Flags)

Parameter:

SFileSrc	: Quelldatei (z.B. ein schwarzes Rechteck)
SFileTgt	: Zieldatei in die die Quelldatei einkopiert werden soll
IPageSrc	: Seite der Quelldatei (wichtig für mehrseitige Dokumente)
IPageTgt	: Seite der Zieldatei
LTransp	: Transparente Farbe, diese wird im Quelldokument durchsichtig
X,Y	: Zielkoordinaten
Flags	: Bit 0: 1=opaque Zeichnen, 0=transparent Zeichen

Rückgabewerte:

1: Ok
-2: Fehler beim kopieren der Bildbereiche

Verfügbar seit:

4.00.094

Beispiel:

```
Set Elo=CreateObject("ELO.office")
Id=Elo.GetEntryId(-1)
if Id>1 then
  Src=Elo.GetPostDir & "elo.tif"
  Dst=Elo.GetDocumentPath( Id, 1 )
  MsgBox Id & " : " & Dst & " : " & Src

  MsgBox Elo.MergeImagesEx( Src, Dst, 1, 1, &Hffffff, 100, 200, 1 )
  call Elo.UpdateDocument( Id, 0, Dst )
end if
```

Funktion MergePostPages

Diese Funktion dient zum Verschränken von Seiten innerhalb der Postbox. Sie entspricht der Funktion ‚Seiten verschränken' im Kontextmenü. Die Funktion bearbeitet die jeweils markierten Einträge.

Rückgabewerte:

-2: Keine Einträge in der Postbox selektiert
-1: Kein Arbeitsbereich aktiv

siehe auch: SelectPostBoxLine
UnselectPostBoxLine
PostBoxLineSelected
SelectAllPostBoxLines
UnselectAllPostBoxLines

Property MinDocLevel (int)

Über dieses Property kann der Ablagelevel für Dokumente gesetzt oder gelesen werden. Somit können je nach Einstellung Dokumente in jeder Ebene abgelegt werden.

Der schreibende Zugriff auf das Property ist erst ab der Version 3.00.420 möglich. Beachten Sie dabei bitte, dass eine Änderung dieses Werts lokal auf diesen Client beschränkt bleibt und auch nicht abgespeichert wird.

Werte :

1	Schrank
2	Ordner
3	Register
...	
253	Register
254	Dokument

siehe auch:

Funktion MovePostboxFile

Funktion MovePostboxFile2

Verschiebt oder kopiert eine Datei aus der eigenen Postbox in die eines anderen Anwenders (Parameter iUser). Mode=0 heißt verschieben, Mode=1 heißt kopieren. Das Bit mit der Wertigkeit 2 bestimmt, ob bei Fehlersituationen eine MessageBox angezeigt wird oder nicht. Der normale MovePostbox Befehl wird nicht im Postboxreport aufgeführt, nur Aktionen über MovePostbox2 werden protokolliert.

int MovePostboxFile(AnsiString sDataFile, int iUser, int Mode)
int MovePostboxFile2(AnsiString sDataFile, int iUser, int Mode,AnsiString sReportParam)

Parameter:

sDataFile	Dateiname
iUser	Anwendernummer des Empfängers
iMode	0=verschieben
	1=kopieren
sReportParam	Zusätzlicher Parameter für den Postboxreport (Kurzbezeichnung).

Rückgabewerte:

1: Datei erfolgreich verschoben/kopiert
-1: Fehler beim Verschieben/Kopieren

siehe auch: ActivePostFile
FindUser

Funktion MoveToArchive

Funktion MoveToArchiveEx

Mit Hilfe dieser Funktion können Sie den aktiven Postboxeintrag (über AddPostboxFile entstanden) in das Archiv übertragen lassen.

Der ObjektIndex kann verschiedene Formen aufweisen:

Eintrag in die Chaosablage (nur für Dokumente erlaubt):

Das Property ObjIndex enthält einen leeren Eintrag (ObjIndex=""). Es wird also kein Ablageort vorgegeben, das Dokument erscheint also nicht in der Aktenstruktur, es kann nur über die Recherche angezeigt werden.

Eintrag über eine interne ELO Objekt-ID:

ObjIndex enthält die ObjektId des Vorgängerknotens (ObjIndex="#12345"), angeführt von dem Symbol #. Es liegt in Ihrer Verantwortung sicherzustellen, daß dieser Vorgängerknoten existiert und vom richtigen Typ ist.

Eintrag über einen Zugriffspfad:

ObjIndex enthält einen Zugriffspfad auf den Vorgängerknoten, beginnend mit dem Symbol ¶ (ObjIndex="¶Schrank¶Ordner¶Register"). Dieser Pfad setzt sich aus den Kurzbezeichnungen, getrennt durch das Symbol ¶ („¶"=Alt 0182), zusammen. Achten Sie bitte darauf, daß bei dieser Vorgehensweise innerhalb von einer Ebene nicht zweimal der gleiche Begriff auftreten darf, da sonst keine eindeutige Zuordnung erfolgen kann. Im Ordner Rechnungen darf also nicht zweimal ein Register März auftreten. Allerdings kann das Register März in jedem beliebigen anderen Ordner verwendet werden.

Eintrag über einen Schlüsselbegriff (Nur für Dokumente):

Sie können in einem Register bis zu drei Schlüsselbegriffe hinterlegen (z.B. KDNR 123). Wenn Sie in ObjIndex dann den Text KDNR=123 hinterlegen, wird das oben genannte Register als Vorgängerknoten verwendet.

int MoveToArchive(AnsiString ObjektIndex)
int MoveToArchiveEx(AnsiString ObjektIndex, AnsiString VersionNo, AnsiString VersionComment)

Parameter:

ObjektIndex	Ablageziel in der Aktenstruktur
VersionNo	Versionsnummer für die Versionsgeschichte des ersten Dokuments
VersionComment	Versionskommentar

Rückgabewerte:

-3: Fehler beim Übertragen aus der Postbox in das Archiv
-2: Unbekanntes oder fehlendes Ablageziel
-1: kein Arbeitsbereich aktiv
1: ok

siehe auch: LookupIndex
AddPostboxFile

Property ObjAcl (AnsiString)

Über das Property ObjAcl können Sie die AccessControlList des aktuellen Eintrags abfragen oder setzen. Dabei ist für die Abfrage mindestens ein lesender Zugriff auf das Objekt notwendig, für das Setzen ein schreibender.

Wenn Sie das Property abfragen erhalten Sie einen Text der Form <Eintrag>,<Eintrag>,...<Eintrag>
Unter Eintrag steht erst mal ein Kennzeichen um was für ein Zugriffsrecht es sich handelt und anschließend die Nummer des betroffenen Schlüssels, Anwenders oder Gruppe. Das Kennzeichen ist immer mindestens einem Zeichen, folgende Möglichkeiten existieren:

- K Es handelt sich um einen Schlüsseleintrag
- R Ein Anwender- oder Gruppeneintrag mit Leserecht
- W Ein Anwender- oder Gruppeneintrag mit Schreibrecht
- D Ein Anwender- oder Gruppeneintrag mit Löschrecht
- E Ein Anwender- oder Gruppeneintrag mit Dateibearbeitungsrecht

Die Kennzeichen R, W, D, E können miteinander kombiniert werden, K muss immer alleine mit einer Schlüsselnummer stehen.

Beispiel: Sie erhalten einen Eintrag „K2,R3,RW4,RWDE5“. Dann ist der Schlüssel 2 gesetzt und der Anwender oder die Gruppe 3 hat Leserecht, 4 hat Lese- und Schreibrecht und 5 darf lesen, schreiben, das Dokument löschen und die Dokumentendatei bearbeiten..

Beispiel:

```
Set Elo=CreateObject("ELO.office")
ObjectId=Elo.GetEntryId (-1)
rv=Elo.PrepareObject(ObjectID,0,0)
MsgBox Elo.ObjAcl
if Elo.ObjAcl="" then
  Elo.ObjAcl="RW7"
else
  Elo.ObjAcl=Elo.ObjAcl&",RW7"
end if
Elo.UpdateObject
MsgBox Elo.ObjAcl
```

siehe auch: PromoteAcl

Property ObjFlags (int)

Das Property ObjFlags hat je nach Art des Objekts unterschiedliche Bedeutungen. Falls es sich um einen Schrank, Ordner oder Register handelt, beinhalten die ObjFlags die Sortierreihenfolge der untergeordneten Objekte.

```
#define PLO_MANUAL  0  // manuelle Sortierreihenfolge
#define PLO_ALPHA   1  // Alphabetische Reihenfolge
#define PLO_XDATE   2  // Dokumentendatum (nur bei Registern sinnvoll)
#define PLO_IDATE   3  // Ablage- bzw. Erzeugungsdatum
#define PLO_IXDATE  4  // Invers - Dokumentendatum
#define PLO_IIDATE  5  // Invers - Ablagedatum
#define PLO_IALPHA  6  // Invers - Alphabetisch
```

Handelt es sich bei dem Objekt um ein Dokument, dann beinhalten die Flags die Versionskontrollstufe und den Volltextstatus.

Bit 0,1:
- 00 keine Versionskontrolle, Dokument kann beliebig verändert werden.
- 01 Versionskontrollierte Ablage, alte Versionen bleiben erhalten.
- 10 Dokumentenechte Ablage, keine Bearbeitung mehr möglich.
- 11 reserviert

Bit 6:
- 0 keine Volltextnachbearbeitung.
- 1 zur Volltextnachbearbeitung anmelden.

Alle anderen Bits sind reserviert oder werden für interne Zwecke benötigt, sie sind mit 0 zu belegen.

siehe auch:
- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- MaskFlags

Property ObjGuid (AnsiString)

Das Property ObjGuid enthält die ELO-interne global eindeutige Objektbezeichnung des aktuellen Eintrags. Diese GUID ist für jedes ELO Objekt weltweit eindeutig und bleibt auch bei der Replikation erhalten. Dieses Property kann nur gelesen werden und ist nur dann verfügbar, wenn die Replikation eingeschaltet ist.

Maximale Länge: 32 Zeichen

siehe auch: GetGuidFromObj
GetObjFromGuid

Property ObjIDate (AnsiString)

Das Property ObjIDate enthält das Ablagedatum des Dokuments oder Ablagestrukturelements. Die Datumsangabe muß sich an dem aktuell im System eingestellten Format für Datumseingaben orientieren, im Allgemeinen also in der Form TT.MM.JJJJ vorliegen.

Beachten Sie bitte, dass es nicht sinnvoll ist, das Property vor der Ablage ins Archiv zu setzen (z.B. in der Postbox oder während der Ablage), da dieser Wert vom Client bei der Ablage automatisch mit dem aktuellen Tagesdatum überschrieben wird. Falls Sie den Eintrag aus irgendwelchen Gründen verändern müssen, dann sollten Sie dieses nach der Ablage durchführen.

Maximale Länge: 12 Zeichen

siehe auch: ObjSDate
ObjXDate

Property ObjIndex (AnsiString)

Das Property ObjIndex enthält den Ablageort des aktuellen Objekts. Dieser Ablageort muß „Typgerecht“ ausgewählt werden, sie dürfen für einen Ordner also nur einen Schrank als Ablageort auswählen, kein Register oder einen anderen Ordner.

Dieser ObjektIndex kann verschiedene Formen aufweisen:

Eintrag in die Chaosablage (nur für Dokumente erlaubt):

Das Property ObjIndex enthält einen leeren Eintrag (ObjIndex=““). Es wird also kein Ablageort vorgegeben, das Dokument erscheint also nicht in der Aktenstruktur, es kann nur über die Recherche angezeigt werden.

Eintrag über eine interne ELO Objekt-ID:

ObjIndex enthält die ObjektId des Vorgängerknotens (ObjIndex=“#12345“), angeführt von dem Symbol #. Es liegt in Ihrer Verantwortung sicherzustellen, daß dieser Vorgängerknoten existiert und vom richtigen Typ ist.

Eintrag über einen Zugriffspfad:

ObjIndex enthält einen Zugriffspfad auf den Vorgängerknoten, beginnend mit dem Symbol ! (ObjIndex=“¶Schrank¶Ordner¶Register“). Dieser Pfad setzt sich aus den Kurzbezeichnungen, getrennt durch das Symbol ¶ („¶“=Alt 0182), zusammen. Achten Sie bitte darauf, daß bei dieser Vorgehensweise innerhalb von einer Ebene nicht zweimal der gleiche Begriff auftreten darf, da sonst keine eindeutige Zuordnung erfolgen kann. Im Ordner Rechnungen darf also nicht zweimal ein Register März auftreten. Allerdings kann das Register März in jedem beliebigen anderen Ordner verwendet werden.

Eintrag über einen Schlüsselbegriff (Nur für Dokumente):

Sie können in einem Register bis zu drei Schlüsselbegriffe hinterlegen (z.B. KDNR 123). Wenn Sie in ObjIndex dann den Text KDNR=123 hinterlegen, wird das oben genannte Register als Vorgängerknoten verwendet.

Eintrag über *Schlüsseltext (???)

Maximale Länge: 200 Zeichen

Siehe auch: LookupIndex
ObjMName

Property ObjInfo (int)

Das Property ObjInfo enthält einen vom Schnittstellenprogrammieren frei verwendbaren Wert. Bei einem Importvorgang wird hier die ObjId aus der alten Archivposition gespeichert, bei einem automatischen Archivabgleich Office-Professional wird in der Office-Version hier die ObjId aus dem Professional Zentralarchiv hinterlegt.

siehe auch: ObjShort
ObjMemo
ObjIDate
ObjXDate
ObjSReg

Property ObjKey (int) (invalid)

Das Property ObjKey setzt oder liest den Schlüssel eines Eintrags.

siehe auch: ObjShort
ObjMemo
ObjIDate
ObjXDate
ObjSReg

Property ObjLock(int) read only

Das Property ObjLock liest die Sperre des aktuellen Eintrags. Hierbei steht eine –1 für keine Sperre, alle anderen Werte geben die Anwendernummer des Eigentümers der Sperre an.

Beispiel:

```
set Elo = CreateObject("ELO.office")

id=Elo.GetEntryId(-1)
if id>1 then
  Elo.PrepareObjectEx id,0,0
  LockUser=Elo.ObjLock
  if LockUser<0 then
    MsgBox "keine Anwendersperre"
  else
    MsgBox "Gesperrt durch " & Elo.LoadUserName(LockUser)
  end if
end if
```

siehe auch: ObjShort
ObjMemo
ObjIDate
ObjXDate
ObjSReg

Property ObjMainParent (int)

Das Property ObjMainParent bestimmt den Hauptvorgänger eines Ordners, Registers oder Dokuments. Achtung: wenn dieses Property verändert wird, muß sichergestellt werden, daß der neue Hauptvorgänger eines Ordners immer ein Schrank ist, der Hauptvorgänger eines Registers immer ein Ordner ist und der Hauptvorgänger eines Dokuments immer ein Register ist. Fehler können zu undefinierten Verhalten innerhalb von ELO führen (z.B. kann dann die Funktion GotoObject nicht mehr korrekt ausgeführt werden).

siehe auch: DocKey
DocKey
DocKind
DocPath

Property ObjMaskNo (int)

Das Property ObjMaskNo setzt oder liest den aktuell eingestellten Dokumententyp.

siehe auch: ObjFlags

Property ObjMemo (AnsiString)

Über dieses Property können Sie den Memo Text für das aktuelle Objekt setzen. Dieser Memo Text enthält bei Bedarf eine allgemeine Beschreibung zu dem Eintrag und ist bei allen Objekt- und Dokumententypen verfügbar.

Maximale Länge: 30000 Zeichen (incl. ObjMemoInfo)

siehe auch: ObjShort
ObjFlags
IDate
Xdate
ObjMemoInfo

Property ObjMemoInfo (AnsiString)

Über dieses Property können Sie im Memo-Feld unsichtbaren Text für das aktuelle Objekt setzen. Dieser MemoInfo Text kann nur über die Automation Schnittstelle gelesen oder geschrieben werden und ist bei allen Objekt- und Dokumententypen verfügbar.

Maximale Länge: 30000 Zeichen (incl. ObjMemo)

siehe auch: ObjShort
ObjFlags
IDate
Xdate
ObjMemo

Property ObjMName (AnsiString)

Liest/Schreibt die Kurzbezeichnung der aktuell aktiven Maske (Dokumententyp). Dieses Feld wird beim Lesen oder Erzeugen eines Objektes (PrepareObject) aus der Maskendefinitionstabelle gesetzt. Beachten Sie, daß durch eine Änderung dieses Eintrags keine Änderung des Dokumententyps (wurde bei PrepareObject „für alle Zeiten“ festgelegt) vornehmen.

Maximale Länge: 40 Zeichen

siehe auch: ReadObjMask
WriteObjMask

Property ObjOwner (int)

Das Property ObjOwner gibt Ihnen die Möglichkeit den Eigentümer eines Eintrags zu ermitteln oder zu verändern..

siehe auch: ObjInfo

Property ObjSDate (AnsiString), ObjSIDate, ObjSVDate

Das Property ObjSDate enthält ein zweites Dokumentendatum für die Recherche eines Datumsbereichs. Die Datumsangabe muß sich an dem aktuell im System eingestellten Format für Datumseingaben orientieren, im Allgemeinen also in der Form TT.MM.JJJJ vorliegen. Über ObjXDate wird das Startdatum, über ObjSDate das Endedatum des Suchbereichs bestimmt. ObjSDate behandelt das zweite Dokumentendatum, ObjSIDate das zweite Ablagedatum und ObjSVDate das zweite Verfallsdatum. Diese Einträge werden nur von Recherchemasken gefüllt und sind bei Archivobjekten undefiniert

Maximale Länge: 12 Zeichen

siehe auch: ObjIDate
ObjXDate

Property ObjSReg (AnsiString)

Das Property ObjSReg enthält die Kurzbezeichnung eines Registers auf dem Registertab. Bei allen anderen Objekten wird dieser Wert zwar akzeptiert und gespeichert aber innerhalb von Elo nicht angezeigt oder zur Bearbeitung angeboten.

Maximale Länge: 8 Zeichen

siehe auch: ObjInfo

Property ObjShort (AnsiString)

Über das Property ObjShort können Sie die Kurzbezeichnung des aktuellen Objektes lesen oder schreiben. Dieser Text sollte (muß) für jedes Objekt eingetragen werden, da er in der Aktenstruktursicht die einzige Orientierung für den Anwender darstellt.

Maximale Länge: 50 Zeichen

siehe auch: ObjMemo
ObjFlags

Property ObjStatus (int)

Über das Property ObjStatus können Sie ermitteln, ob ein Objekt gelöscht ist (Status <>0). Verwenden Sie dieses Property nicht zum Löschen von Einträgen indem Sie es einfach mit einer 1 beschreiben.

siehe auch: ObjInfo

Property ObjType (int) (invalid)

Property ObjTypeEx (int)

Über das Property ObjType können Sie den Objekttyp ermitteln. Bei Archiven mit einer vierstufigen Hierarchie wird mit ObjType gearbeitet, bei Archiven mit mehr als 4 Hierarchiestufen mit ObjTypeEx.

ObjType
Es stehen folgende Werte zur Verfügung

1: Schrank
2: Ordner
3: Register
4: Dokument

ObjTypeEx
Es stehen folgende Werte zur Verfügung

1: Schrank
2: Ordner
3:...
.
253: Register
254: Dokument

Bei der Veränderung des Objekttyps ist äußerste Vorsicht angebracht. Im Normalfall gibt es keinen Grund dazu hier irgendetwas zu verändern. Der Objekttyp wird bereits mit der Erzeugung des Eintrags entsprechend der Lage gesetzt und kann deshalb nicht mehr sinnvoll verändert werden. Versuchen Sie nicht, hier durch Manipulationen Register in Registern anzulegen, dieses Tun führt unweigerlich zu Inkonsistenzen in der Datenbank.

siehe auch: ObjKey
ObjFlags
ObjKind

Property ObjVDate (AnsiString)

Verfügbar ab:	**Ver 3.00.228**

Das Property ObjVDate enthält das Dokumentenverfallsdatum. Die Datumsangabe muss sich an dem aktuell im System eingestelltem Format für Datumseingaben orientieren, im Allgemeinen also in der Form TT.MM.JJJJ vorliegen. Wenn Sie eine Suche nach einem Verfallsdatumsbereich durchführen wollen, dann geben Sie den Bereich in ObjVDate (von) und ObjSVDate (bis) ein.

Maximale Länge: 12 Zeichen

siehe auch: ObjSDate
ObjXDate
ObjIDate
ObjSVDate

Property ObjXDate (AnsiString)

Das Property ObjXDate enthält das Dokumentendatum (Bei Rechnungen z.B. das Rechnungsdatum). Die Datumsangabe muß sich an dem aktuell im System eingestellten Format für Datumseingaben orientieren, im Allgemeinen also in der Form TT.MM.JJJJ vorliegen.

Maximale Länge: 12 Zeichen

siehe auch: ObjIDate
ObjSDate

Funktion OcrAddRect

Fügt ein weiteres Rechteck in die OCR Erkennungs-Rechteckliste ein. Zum Aufbau einer Rechteckliste sollte immer zuerst mit OcrClearRect ein definierte Zustand hergestellt werden, danach kommt eine Folge von maximal 32 OcrAddRect Befehlen. Das Rechteck wird in Form eines Strings mit den linken, oberen Koordinaten und den rechten, unteren Koordinaten angegeben. Alle Werte werden in Promille ausgedrückt (z.B.: 0,0,999,999 ist die ganze Seite).

int OcrAddRect(AnsiString Rechteck)

Parameter: Rechteck in der Form xa,ya,xe,ye

Rückgabewerte: -1: Fehlerhafte Rechteckliste
1: ok

Funktion OcrAnalyze und OcrAnalyzeEx

Die Funktion OcrAnalyze übergibt den Namen der auszuwertenden Datei und liest anhand der voreingestellten Rechteckliste die Texte in die Textliste ein.

int OcrAnalyze(AnsiString Dateiname, int Seitennummer)
int OcrAnalyzeEx(AnsiString Dateiname, int Seitennummer, int Mode)

Parameter	Dateiname:	Name der zu analysierenden Datei
	Seitennummer:	Zu untersuchende Seite (erste Seite ist 0)
	Mode:	1: Nur Ziffern erkennen, 0: Alle Zeichen erkennen

Rückgabewerte:	-1:	Fehler bei der OCR Erkennung
	-2:	OCR Engine konnte nicht initialisiert werden
	1:	ok

Funktion OcrClearRect

Löscht die interne Rechteckliste der OCR Erkennung. Dieser Schritt sollte als Vorbereitung vor dem Setzen einer neuen Rechteckliste ausgeführt werden, so daß ein definierter Stand existiert.

int OcrClearRect()

Parameter: keine

Rückgabewerte: immer 1, ok

Funktion OcrGetPattern

Liefert einen erkannten Teiltext eines Musters zurück. Beachten Sie, dass jeder Teil eines Musters einen Teiltext besitzt, auch wenn er eigentlich fest oder gar leer ist. Das Muster „*'Rechnung'_N*“ besitzt als ersten Musterteil ein * (beliebige Zeichenfolge), danach folgt als zweiter Teil der feste Text 'Rechnung', der dritte Teil ist eine Folge von Leerzeichen (die auch leer sein kann), als vierter Teil folgt eine Nummer und zuletzt als fünfter Teil kommt wieder ein beliebiger Teiltext, der komplette Rest nach der erkannten Nummer (kann auch wieder leer sein). Beachten Sie bitte, daß Sie den ersten Teiltext mit OcrGetPattern(0) abrufen müssen (also mit 0, nicht mit 1 beginnend).

AnsiString OcrGetPattern (int PatternNo)

Parameter: PatternNo: Nummer des zu liefernden Teiltextes aus dem letzten Muster

Rückgabewerte: Erkannter Text (im Fehlerfalle ein Leerstring)

Funktion OcrGetText

Nach der OCR Analyse stehen in einem Textfeld die erkannten Teile der Rechecklіste zur Verfügung. Über den Befehl OcrGetText können Sie diese Texte auslesen, OcrGetText(0) liefert den Text zum ersten Rechteck, OcrGetText(1) zum zweiten usw..

AnsiString OcrGetText(int TextNo)

Parameter: TextNo: Text zum n. Eintrag der Rechteckliste (erster Eintrag=0)

Rückgabwerte: Erkannter Text (im Fehlerfall wird ein Leerstring übergeben).

Funktion OcrPattern

Die Funktion übernimmt ein Muster und einen Eingabetext und trennt diesen gemäß des vorgebenen Musters in Teiltexte auf. Den Aufbau des Musterstrings können Sie dem Anhang entnehmen. Der Parameter PrepareText bestimmt über die Vorverarbeitung des Eingangstextes vor der Mustererkennung (um überflüssige white spaces (Leerzeichen etc.) zu entfernen). Hierbei stehen folgende Kombinationen zur Verfügung:

- 0: Es werden keine WS im Text entfernt
- 1: Mehrere WS werden auf einen verkürzt
- 2: Es werden alle WS aus dem Text entfernt

Zusätzlich können Sie noch folgende Werte auf diesen Startwert aufaddieren:

- 8: Es werden alle WS am Textanfang und Textende entfernt (entspricht in etwa dem TRIM() Befehl in BASIC
- 16: Es werden nach jedem Zeilenwechsel die WS am Zeilenanfang entfernt

int OcrPattern(int PrepareText, AnsiString Muster, AnsiString Text)

Parameter:	PrepareText	Quelltextvorverarbeitung
	Muster	zu prüfendes Muster
	Text	zu prüfender Text

Rückgabewerte:	-1:	Fehler bei der Vorbereitung des Textes aufgetreten
	-2:	Fehler bei der Aufbereitung des Musters aufgetreten
	-3:	Das Muster konnte im Text nicht gefunden werden
	>0:	Das Muster wurde gefunden, der Rückgabewert enthält die Anzahl der Teiltexte.

Property OfficeMaskNo (AnsiString)

Das Property OfficeMaskNo enthält die Konfigurationseinstellung des ELO Clients "Systemverwaltung" – "Optionen" – "Allgemein" – "Standard-Ablagemasken für neue Einträge" – "Microsoft Office Dokumente". Dieses Property kann nur gelesen werden.

Property OkEnabled (int)

Mit dem Property OkEnabled kann geprüft werden, ob der Ok-Button im ELO Verschlagwortungsdialog aktiviert ist und der Anwender Änderungen an den Indexdaten vornehmen darf. Das Property muss abgeprüft werden, wenn der ELO-eigene Verschlagwortungsdialog durch einen eigenen Dialog (z.B. eine Visual Basic Maske) ersetzt wird. Nur wenn der der Rückgabewert den Wert 1 besitzt, kann durch Setzen des Properties ScriptActionKey auf den Wert 10 ein Ok-Click an ELO durchgemeldet werden. Es empfiehlt sich also, den Ok-Button der eigenen Maske nur dann anklickbar zu machen, wenn OkEnabled den Wert 1 hat.
Dieses Property kann nur gelesen werden.

Werte:

0: Ok-Button nicht aktiviert
1: Ok-Button aktiviert

Property (AnsiString) OpenSave (int Wert)

Über dieses Property werden die Werte für die Open & Save Dialogbox gesetzt bzw. gelesen.

Wert	Objekt	Eingabe		R/W
1	DefaultExt	Text 3 Stellen (Rest wird abgeschnitten)		R/W
2	FileEditStyle	**Wert**	**Bedeutung**	R/W
		Edit	Edit Box	
		ComboBox	Combo Box	
3	FileName	Text : <Dateiname mit Pfad>		R/W
4	Files	Anzahl der ausgewählten Dateien bei Multiselect		R/-
5	Filter	Text : Text files (*.txt)|*.TXT| C++ files (*.cpp)|*.CPP		R/W
6	FilterIndex	Zahl :1 bis Anzahl Filter Einträge		R/W
7	InitialDir	Text : <Pfad mit Laufwerksangabe>		R/W
8	Options	Siehe unten (Verknüpfen mit |)		R/W
9	Title	Text : <Titel>		R/W
100 … …200	FileName	Ausgewählte Dateinamen, wenn die Option AllowMultiSelect mit angegeben wurde und mehrere Dateien gewählt sind		R/-

Options:

Value	Meaning
AllowMultiSelect	Allows users to select more than one file in the dialog.
CreatePrompt	Generates a warning message if the user tries to select a nonexistent file, asking whether to create a new file with the specified name.
ExtensionDifferent	This flag is turned on at runtime whenever the selected filename has an extension that differs from
DefaultExt.	If you use this flag an application, remember to reset it.
FileMustExist	Generates an error message if the user tries to select a nonexistent file.
HideReadOnly	Removes the Open As Read Only check box from the dialog.
NoChangeDir .	After the user clicks OK, resets the current directory to whatever it was before the file-selection dialog opened
NoDereferenceLinks	Disables dereferencing of Windows shortcuts. If the user selects a shortcut, assigns to FileName the path and file name of the shortcut itself (the .LNK file), rather than the file linked to the shortcut.
NoLongNames	Displays 8.3-character file names only.
NoNetworkButton	Removes the Network button (which opens a Map Network Drive dialog) from the file-selection dialog. Applies only if the ofOldStyleDialog flag is on.
NoReadOnlyReturn	Generates an error message if the user tries to select a read-only file.
NoTestFileCreate	Disables checking for network file protection and inaccessibility of disk drives. Applies only when the user tries to save a file in a create-no-modify shared network directory.
NoValidate	Disables checking for invalid characters in file names. Allows selection of file names with invalid characters
OldStyleDialog	Creates the older style of file-selection dialog.
OverwritePrompt	Generates a warning message if the user tries to select a file name that is already in use, asking whether to overwrite the existing file.
PathMustExist	Generates an error message if the user tries to select a file name with a nonexistent directory path.
ReadOnly	Selects the Open As Read Only check box by default when the dialog opens.
ShareAware	Ignores sharing errors and allows files to be selected even when sharing violations occur.

ShowHelp	Displays a Help button in the dialog.

Bsp.:

Setzen der Default Dateierweitung auf EXE
ELO. OpenSave (1) = „EXE“

Einlesen des Übergebenen Dateinamens
Filename = ELO. OpenSave (3)

siehe auch: OpenSaveDialog

Funktion (AnsiString) OpenSaveDialog (int Typ)

Diese Funktion erstellt einen Open- oder Save Dialog je nach Typ. Die Werte für Titel, FileName etc. Sind über das Property OpenSave erreichbar.

Wert	**Funktion**
1	Open Dialog
2	Save Dialog

Rückgabewert :
-1 – Nicht unterstützter Typ
0 – Dialog mit Abbrechen verlassen
1 – Dialog mit OK verlassen (Property OpenSave wird aktualisiert)

Bsp.:

Erstellt einen Open Dialog
ELO. OpenSaveDialog (1)

Erstellt einen Save Dialog
ELO. OpenSaveDialog (2)

Werte müssen vorher mit dem Property OpenSave zugewiesen werden. Ansonsten erscheint der Dialog mit Windows default Werten.

siehe auch: OpenSave

Funktion OrientFile

Mit dieser Funktion können Sie eine Bilddatei in der Postbox analysieren lassen, eine eventuelle Rotation um 90, 180 oder 270 Grad wird korrigiert. Die Analyse erfolgt unter Verwendung des OCR-Systems. Vor Verwendung der Funktion muss die Funktion OCRInit aufgerufen werden, nach Beendigung die Funktion OCRExit. Die Funktion arbeitet auch mit Multipage TIFF-Dateien.

Als Dateiname kann ein Eintrag aus der Postbox übergeben werden (ohne Pfad, nur der Dateiname) oder ein Index in die Postliste (durch ein # gekennzeichnet, z.B. #0 ist der erste Eintrag in der Postliste).

int OrientFile(AnsiString FileName)

Parameter:

FileName: Name der zu untersuchenden Datei

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Kein Dateiname
-3: Fehler beim Laden der Datei
-4: Fehler beim Laden der Seite (OCR)
-5: Fehler bei OCR
-6: Fehler beim Abspeichern der Datei
1: Ok

Siehe auch: GetDocumentOrientation
RotateFile
UpdatePostbox

Property OutlookName (AnsiString)

Dieses Property gibt den aktuellen Namen der Person (Gruppe) wieder, an die die Wiedervorlage geht.

siehe auch: DelOutlookName

PopupObjID

Wird die Funktion "ClickOn" mit einer Funktion verwendet, die das Kontextmenü betrifft, ist es vom Vorteil vorab die Funktion "PopupID" auf das entsprechende Objekt anzuwenden.
Elo.PopupObjID = lngObjID

Funktion PostBoxLineSelected

Mit dieser Funktion kann getestet werden, ob eine Zeile in der Postbox selektiert ist.

int PostBoxLineSelected(int LineNo)

Parameter:

LineNo	zu prüfende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
0: Zeile nicht selektiert
1: Zeile selektiert

siehe auch: SelectPostBoxLine
UnselectPostBoxLine
SelectAllPostBoxLines
UnselectAllPostBoxLines

Funktion PrepareObject (invalid)

Funktion PrepareObjectEx

Diese Funktion liest ein ELO Objekt (Schrank, Ordner, Register oder Dokument) in einen internen Editierpuffer. Von hier aus können die verschiedenen Felder wie z.B. Kurzbezeichnung, Farbe oder Memotext gelesen, bearbeitet und verändert werden. Diese Änderungen können dann mit UpdateObject permanent in die Datenbank übergeben werden.

Wenn Sie ein neues Objekt anlegen wollen, müssen Sie auch hier zuerst über diesen Aufruf mit der ObjectId 0 einen neuen, leeren Eintrag initialisieren lassen. Wenn Sie einen bestehenden Eintrag als ‚neu' markieren wollen (d.h. alle Daten bleiben erhalten, beim Speichern wird aber ein neuer Eintrag erzeugt), müssen Sie eine –2 als ObjectId angeben. Diese Funktion ist erst ab Version 2.02.059 verfügbar.

Als dritte Möglichkeit kann noch ein Postbox-Eintrag als „aktiver Postboxeintrag" eingerichtet werden. In diesem Falle geben Sie eine –1 als ObjectId und die Postbox-Zeilennummer (beginnend mit 0) als ObjectType an.

Bei Archiven mit einer vierstufigen Hierarchie wird mit PrepareObject gearbeitet, bei Archiven mit mehr als 4 Hierarchiestufen mit PrepareObjectEx.

Beachten Sie bitte, dass beim Lesen von Postboxeinträgen die Fehlerzustände –5 und –7 zurückgegeben werden können. Das sind nur Fehler im Sinne von „die Verschlagwortung konnte nicht gelesen werden da sie noch nicht angelegt wurde". Der Postboxeintrag existiert aber unverschlagwortet und kann bearbeitet werden.

int PrepareObject(int ObjectId, int ObjType, int MaskNo)

Parameter:

ObjectId	0: für einen neuen Eintrag, -1: für einen Postbox-Eintrag, -2: aktueller Eintrag wird als ‚Neu' markiert ansonsten: Interne Elo-Zugriffsnummer auf das Objekt.
ObjectType	1=Schrank, 2=Ordner, 3=Register, 4=Dokument, falls Postboxeintrag: Zeilennummer (0..n)
MaskNo	Dokumententyp (siehe auch ReadObjMask)

int PrepareObjectEx(int ObjectId, int ObjType, int MaskNo)

Parameter:

ObjectId	0: für einen neuen Eintrag, -1: für einen Postbox-Eintrag, ansonsten: Interne Elo-Zugriffsnummer auf das Objekt.
ObjectType	1=Schrank, 2=Ordner, 3=..., ..., 253=Register, 254=Dokument, falls Postboxeintrag: Zeilennummer (0..n)
MaskNo	Dokumententyp (siehe auch ReadObjMask)

Rückgabewerte:

-1: Unbekannter Maskentyp
-2: Fehler beim Lesen des Eintrags
-3: Kein Arbeitsbereich aktiv

-4: Kein Postboxpfad gefunden
-5: Der Postboxeintrag besitzt noch keine Verschlagwortung und ist nicht selektiert
-6: Postboxeintrag nicht gefunden
-7: Der Postboxeintrag besitzt noch keine Verschlagwortung und ist selektiert
-8: Kein Leserecht
1: Neuer leerer Eintrag steht zur Verfügung
2: Bestehenden Eintrag aus der Datenbank gelesen
3: Bestehenden, selektierten Postboxeintrag gelesen
4: Bestehenden, nicht selektierten Postboxeintrag gelesen
5: Bestehenden, gelöschten aber noch nicht dauerhaft entfernten Eintrag gelesen

Beispiel: erstes Postboxdokument aufgreifen, mit der Maskennummer 2 belegen und die Kurzbezeichnung sowie das erste Indexfeld automatisch füllen. Anschließend wird das Dokument auf dem vorgegeben Pfad ins Archiv übertragen:

```
' Zielregister für das Dokument
RegisterId = "¶Schrank¶Ordner¶Register"
MaskNo = 2
Set Elo=CreateObject("ELO.office")

' Zur Sicherheit erst die Postbox aktualisieren
Elo.UpdatePostbox

x=Elo.PrepareObjectEx( -1, 0, MaskNo )
if x>0 or x=-5 or x=-7 then
  Elo.ObjShort="Test" & Time
  call Elo.SetObjAttrib(0,"Index 1")
  call Elo.AddPostboxFile("")
  x=Elo.MoveToArchive( RegisterId )
else
  MsgBox "Kein Dokument in der Postbox vorgefunden"
end if
```

siehe auch: UpdateObject
LookupIndex

Funktion PromoteAcl

Mit dieser Funktion können Sie die Untereinträge eines Objektes an die aktuelle Acl anpassen. Der Aufbau der ACL Einträge ist beim Property ObjAcl beschrieben.

int PromoteAcl (int iObjId, int iShowResult, int iExact, int UpdateVal, AnsiString mAcl, AnsiString mAdd, AnsiString mDel)

Parameter:

iObjId	Startknoten der umzustellenden Einträge
iShowResult	0: kein Ergebnisdialog anzeigen, 1: Ergebnisdialog anzeigen
iExact	0: mAdd und mDel ACLs enthalten die Veränderungen der alten ACL 1: mAcl enthält die neue ACL
UpdateVal	zu beachtende Ebenen Bit 0= Strukturelemente, Bit 1= Dokumente
mAcl	neue AccessControlList (ACL), wird für alle Untereinträge verwendet wenn iExcat=1 ist
mAdd	zusätzliche Einträge für die ACL, wird verwendet wenn iExact=0 ist
mDel	zu entfernende Einträge aus der ACL, wird verwendet wenn iExact=0 ist

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: beendet

siehe auch: ObjAcl

Funktion (int) PrintDocList (AnsiString DokTitel, AnsiString ObjIdList, AnsiString ObjList, int Typ)

Diese Funktion druckt eine Übersicht der angegebenen Objekte.
Wenn der Typ 1 angegeben wird, so können die Spaltenbreiten über das Property PrintDocListTabs eingestellt werden. Standardmäßig sind die Werte auf DIN A4 Portrait eingestellt. Spalten mit einer Breite von 0 werden nicht gedruckt. Sollte der Text über die angegebene Breite gehen wird er automatisch gekürzt und mit 3 Punkten am Ende versehen.

DokTitel : Überschrift (wenn “” übergeben wird ist der Titel : Suchergebnis

ObjIdList : Liste mit ObjektIds getrennt durch den Separator. Nicht vorhandene ObjektIds werden ignoriert.

ObjList : Liste getrennt durch den Separator. Falsche Eingaben werden ignoriert.

Short	- Kurzbezeichnung
Date	- Ablagedatum
Ddate	- Dokumentendatum
Index	- Indexzeilen
Memo	- Memo text

Typ : 32 Bit Integer Maske

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	03	02	01	00

00	0 – Listendarstellung 1 – Spaltendarstellung
01	0 – Kurze Überschriften 1 – Lange Überschriften
02	0 – Keine Trennlinie 1 – Trennlinie nach jedem Eintrag
03	0 – Keinen Dialog anzeigen 1 – Dialog anzeigen (übergebene Werte werden eingesetzt)
x	Momentan unbenutzt (reserviert für Erweiterungen)

Rückgabewert :

-3	- Dialog wurde mit Abbruch beendet
-2	- Keine Objekte zum Druck vorhanden
-1	- Kein Arbeitsbereich aktiv
0	- Dokument wurde zum Drucker gesendet. (im Dialog wurde OK gedrückt)

Bsp.:

Druckt eine Liste der angegebenen ObjektIds mit den Feldern Kurzbezeichnung, Ablagedatum und Indexzeilen. Die Überschrift ist Suchergebnis.
ELO. PrintDocList (“”,”10¶11¶52”,“Short¶Date¶Index“,0)

Druckt eine Tabelle der angegebenen ObjektIds mit den Feldern Kurzbezeichnung, Ablagedatum und Indexzeilen. Die Überschrift ist Meine Suche.
ELO. PrintDocList (“Meine Suche”, ”10¶11¶52”, “Short¶Date¶Index“,1)

siehe auch: PrintDocListTabs

Property PrintDocListTabs (int Nr)

Über diese Property können die Tabulatoren für die Spaltendarstellung des Ausdrucks über PrintDocList eingestellt werden.

int PrintDocListTabs (int Nr)

Nr. :

0	- Default
1	- Spalte Kurzbezeichnung
2	- Spalte Dokumenten Datum
3	- Spalte Ablage Datum
4	- Spalte Index
5	- Spalte Memo

Die Spaltenbreiten sind in mm anzugeben.
Das Feld mit der Nummer 0 ist zum Setzen der Default Werte.
Folgende Eingaben sind zulässig:

1	- Setzt Portrait Default Werte
2	- Setzt Landscape Default Werte

Bsp.:

Setzt die Defaultwerte für Portrait Druck.
ELO. PrintDocListTabs (0) = 1

siehe auch: PrintDocList

Funktion PrintDocument

Diese Funktion druckt das gerade angezeigte Dokument.

PrintDocument(int MitDialog, String DeviceName, String Port, int FromPage, int ToPage, int Copies)

Parameter:

MitDialog	1: vor dem eigentlichen Ausdruck erscheint der Druckerauswahldialog 0: Druckerauswahldialog wird nicht angezeigt
DeviceName	Name des Druckers, kann auch ein im Netzwerk freigegebener Drucker sein falls DeviceName=““ wird der Standarddrucker wird verwendet
Port	Bezeichnung des Druckerports (nur relevant, wenn ein und derselbe Druckertreiber mehrere Drucker über unterschiedliche Schnittstellen ansteuert) falls Port=““ wird der Standardwert verwendet
FromPage, ToPage	Ausdruck erfolgt von Seite bis Seite, wird bei beiden Werten „0“ übergeben, wird das gesamte Dokument gedruckt
Copies	Anzahl Kopien

Rückgabewerte:

1: ok
-1: ELO steht nicht in der Hauptansicht
-2: es wird derzeit kein Dokument angezeigt
-3: Fehler beim Ausdruck

Funktion QueryOption

Mit der Funktion QueryOption können Sie einzelne Einstellungen des Optionen-Dialogs abfragen.

AnsiString QueryOption(int OptionNo)

Parameter:

OptionNo: Funktionsliste siehe SetOption

Rückgabewerte:

Aktueller Wert der Option oder „ERROR“

Beispiel

```
Set Elo=CreateObject( "ELO.office" )

msg="Options:" & vbcrlf & "----------------------" & vbcrlf
for i=0 to 20
  res=Elo.QueryOption(i)
  if res="ERROR" then
    exit for
  end if
  msg=msg & i & ": " & res & vbcrlf
next

MsgBox msg
```

siehe auch: SetOption

Funktion ReadColorInfo

Mit dieser Funktion können Sie eine Farbdefinition in das aktuelle Elo-Farbobjekt einlesen. Auf die Farbeinstellungen können Sie dann mittels der Properties ColorInfo und ColorName zugreifen.

Wenn Sie eine Liste der verfügbaren Farben benötigen, können Sie die Funktion beginnend mit 0 und gesetztem Bit 0x8000 aufrufen. Es wird dann jeweils der nächste passende Farbwert geladen.

int ReadColorInfo(int ColorNo)

Parameter:

ColorNo: Nummer der einzulesenden Farbdefinition

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Ungültiger Wert für die Farbnummer
1: ok

siehe auch: WriteColorNo
ColorInfo
ColorName

Funktion ReadKey (int)

Diese Funktion liest einen Systemschlüsseleintrag aus und übergibt den Namen.

AnsiString ReadKey (int KeyNo)

KeyNo : Schlüsselnummer beginnend mit 1

Rückgabewert:
Name des Schlüssels oder Leerstring bei ungültiger Schlüsselnummer.

Bsp.:

```
Gibt den Namen des Schlüssels mit der Nummer 1 aus.
MsgBox Elo.ReadKey (1)
```

siehe auch: WriteKey
ChangeKey

Funktion ReadObjMask

Mit dieser Funktion können Sie eine Maske (Dokumententyp) in das interne ELO Objekt einlesen. Hierbei werden die Attribut-Bezeichnungs- und Indexfelder aber nicht die Eingabewertfelder mit den definierten Werten vorbesetzt.

int ReadObjMask(int MaskNo)

Parameter:

MaskNo: Nummer der einzulesenden Dokumententypmaske

Rückgabewerte:

-3: Kein Arbeitsbereich aktiv
-2: Ungültiger Wert für die Maskennummer
-1: Fehler beim Lesen der Datenbank
1: ok

Siehe auch: WriteObjMask
GetObjMaskNo

Funktion ReadSwl

Diese Funktion liest eine Ebene aus dem Teilbaum einer Stichwortliste. Jeder Eintrag einer Stichwortliste wird innerhalb seiner Ebene eindeutig durch eine 2-stellige Buchstabenkombination AA, AB, AC .. ZZ gekennzeichnet. Durch die Verkettung aller Kennzeichen aller Ebenen können Sie jedes Stichwort im Baum eindeutig ansprechen. Beginnend mit dem Punkt für die Wurzel können Sie nun z.B. über „.AAABAC“ in der untersten Ebene den ersten Eintrag (AA), darunter den zweiten (AB) und darunter den dritten (AC) Eintrag löschen. Dabei werden alle Einträg unterhalb der gewählten Ebene zusammen mit den jeweiligen Kennungen, getrennt durch das Trennsymbol, zurückgegeben.

Die Gruppe kennzeichnet die Listenzuordnung zu einer Indexzeile (gleicher Eintrag wie im Gruppenfeld des Maskeneditors).

AnsiString ReadSwl(AnsiString Gruppe, AnsiString Parent, AnsiString Delimiter)

Parameter:

Gruppe	Wählt die Stichwortliste aus
Parent	Pfad auf die zu löschenden Einträge
Delimiter	Trennsymbol

Rückgabewerte:

-1: kein Workspace offen
-2: Fehler beim Speichern des Stichwortes
sonst: Liste der Stichwörter mit jeweiliger Kennzeichnung

Beispiel

```
...
Set Elo=CreateObject("ELO.office")

call Elo.DeleteSwl( "THM", "." )

MsgBox Elo.AddSw( "THM", ".", "1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.2" )
MsgBox Elo.AddSw( "THM", ".AA", "1.3" )
MsgBox Elo.AddSw( "THM", ".", "2" )
MsgBox Elo.AddSw( "THM", ".", "3" )

MsgBox Elo.ReadSwl( "THM", ".", " - " )
MsgBox Elo.ReadSwl( "THM", ".AA", " - " )

call Elo.UpdateSw( "THM", ".AB", "2a" )
MsgBox Elo.ReadSwl( "THM", ".", " - " )
...
```

Funktion ReadUser

Die Funktion ReadUser liest einen Anwenderdatensatz aus der Systemdatei ein. Administratoren können beliebige Anwender lesen und bearbeiten, Subadministratoren können nur eigene Anwender lesen und bearbeiten. Das Auslesen von einigen Daten eines Anwenderdatensatzes ist aber auch normalen Usern gestattet.

Über die UserNo –1 können Sie einen leeren Anwenderdatensatz erzeugen. Mittels der UserNo –2 lesen Sie die kumulierten Rechte des aktuellen Anwenders ein (d.h. das Property UserGroups enthält nicht nur die direkten Gruppen sondern auch alle Untergruppen, das gleiche gilt für UserKeys) (verfügbar ab Version 3.00.546).

Wenn Sie einen neuen Anwender anlegen wollen, dann müssen Sie zuerst eine freie User-Nummer suchen (ein freier Anwender ist ein Anwender ohne Namen). Anschließend können Sie dann die User-Properties füllen und den Datensatz speichern (unbedingt darauf achten, dass die UserNummer korrekt ist. Andernfalls überschreiben Sie evtl. einen bestehenden Anwender).

int ReadUser(int UserNo)

Parameter:

UserNo:	Nummer des einzulesenden Anwenders

Rückgabewerte:

-1	Fehler beim Lesen des Anwenders aus dem AccessManager
-2	Nur Administratoren und Subadministratoren können Anwenderdaten lesen
-3	Ein Subadministrator hat versucht einen fremden Anwender zu lesen
1	Ein leerer Anwenderdatensatz ist über UserNo = -1 oder -2 erzeugt worden
2	Anwenderdatensatz korrekt gelesen

siehe auch: ReadUser
UserGroups
UserParent
UserKeys
UserFlags

Funktion ReadUserProperty

Die Funktion ReadUserProperty liest einen Anwenderzusatz aus einem der 8 Anwenderdatenfelder. Beachten Sie bitte, dass die ersten 4 Felder von ELO reserviert sind (z.B. für die NT-Namensumsetzung). Die Felder 5..8 stehen zur freien Verfügung.

AnsiString ReadUserProperty(int PropertyNo)

Parameter:

PropertyNo: Nummer des zu lesenden Properties

Rückgabewerte:

Ausgelesenes Property oder Leerstring bei Fehler

Beispiel:

```
' AutoStart.VBS  20.03.2002
'-----------------------------------------------------------------
' © 2002 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo-digital.de)
'-----------------------------------------------------------------
' Dieses Skript liest aus den Anwenderdaten ein Start-Ordner oder
' Register aus und verzweigt an die angegebene Stelle. Dabei wird
' pro Archiv ein eigener Pfad eingegeben. Die Definition erfolgt in
' dem ersten anwenderdefinierten Feld in den User-Verwaltung in der
' Form |<Archivname1>¶Schrank¶Ordner|<Archivname2>¶<NochEinSchrank>
' Die einzelnen Archivdefinitionen werden also über ein Pipe (|)
' Symbol getrennt. Jede einzelne Definition beginnt mit dem Archiv-
' namen, gefolgt von einem Archivpfad.
' ----------------------------------------------------------------

Set Elo=CreateObject("ELO.office")

Elo.ReadUser( Elo.ActiveUserId )
Param=Elo.ReadUserProperty(5)
if Param<>"" then
  ArcList=Split( Param, "|" )
  ArcName=Elo.GetArcName & "¶"
  for i=LBound(ArcList) to UBound(ArcList)
    ThisEntry=ArcList(i)
    if Left( ThisEntry, Len(ArcName) )=ArcName then
      ThisEntry=Mid(ThisEntry, Len(ArcName), 255)
      ThisId=Elo.LookupIndex( ThisEntry )
      if ThisId>0 then
        Elo.GotoId(0-ThisId)
      end if
    end if
  next
end if
```

siehe auch: WriteUserProperty
UserGroups

UserParent
UserKeys
UserFlags

Funktion ReadWv

Liest einen Wiedervorlagetermin (bestimmt durch den Parameter WvIdent) in den internen Wv-Speicher ein. Dieser Termin kann dann mit den verschiedenen Property-Funktionen ausgelesen werden. Ein WvIdent 0 bewirkt, daß der interne Wv-Speicher gelöscht wird, dieser Aufruf ist vor dem Anlegen eines neuen Termins notwendig.

int ReadWv(int WvId)

Parameter:

WvId: Nummer des zu lesenden Termins, 0: neuen, leeren Termin anlegen

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehler beim Lesen
1: ok

Siehe auch: WriteWv
DeleteWv
WvIdent
WvParent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Funktion ReloadWv

Diese Funktion aktualisisrt die Liste der Wiedervorlagetermine innerhalb des ELO Hauptfensters.

int ReadWv( )

Parameter:

Keine

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: ok

Siehe auch: WriteWv
DeleteWv
WvIdent
WvParent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Funktion RemoveDocs (int, AnsiString, int)

Über diese OLE-Funktion kann der Aufruf des Dialogs zum Entfernen von Altdokumenten automatisiert werden. Als Parameter wird die Dokumentenpfadeinschränkung, das Grenzdatum im ISO-Format (YYYYMMTT) und ein Mode-Parameter mit den codierten Aktionen übergeben.

Der Mode-Parameter setzt sich aus folgenden Werten zusammen:

Entweder 1,2 oder 3 für die Dokumentenkontrolle. Bei 1 wird das Dokument ohne weitere Kontrolle gelöscht (sehr gefählich, im Normalfall nicht anwenden). Bei einer 2 wird ein Größenvergleich ausgeführt und bei einer 3 wird der Dateiinhalt verglichen (analog zu den Optionen aus dem Dialog.

Mit dem Mode 1,2 oder 3 wird erstmal nur eine Löschkontrolle durchgeführt. Es werden dann noch keine Dateien tatsächlich gelöscht sondern es erfolgt nur eine Rückmeldung darüber, wie viele Dokumente gelöscht würden. Erst wenn Sie auf diesen Wert die Zahl 21840 addieren, dann erfolgt eine tatsächliche Löschung (also 21841, 21842 oder 21843).

Als Rückgabewert erhalten Sie eine Fehlernummer oder einen Löschreport. Der Löschreport enthält ein Folge von kommagetrennten Zahlen:

1. Letzte Dokumentennummer
2. Anzahl der gelöschten Dateien
3. Anzahl der Dateien ohne Backup
4. Anzahl der defekten Einträge
5. Umfang der gelöschten Dokumente (falsche Ausgabe bei mehr als 2 GB).

AnsiString RemoveDoc (int PathId, AnsiString Grenzdatum, int Mode)

PathId :	Zu löschender Pfad (0 = alle Pfade)
Grenzdatum :	Grenzdatum im ISO Format, nur ältere Einträge werden gelöscht
Mode:	1,2 oder 3 für die Löschkontrolle, 21841, 21842, 21843 für das Löschen von Dokumenten

Rückgabewert:

-1,…	- Fehler beim Löschen
-2	- Ungültiges Grenzdatum
-3	- kein Arbeitsbereich aktiv
Sonst	-Löschreport

Bsp.:

```
Elo.RemoveDoc (1, “20050727“, 3)
```

Funktion RemoveRef (int, int)

ELO bietet neben der hierarchischen Baumstruktur (Schrank - Ordner - Register - Dokument) die Möglichkeit, daß Sie weitere Referenzen anlegen können. Ein Dokument „Rechnung Müller“ kann im Register „Rechnungen“ abgelegt werden und mit einer zusätzlichen Referenz im Register „Müller“ eingetragen werden. Obwohl es das Dokument dann nur einmal im System gibt, ist es von beiden Stellen aus sichtbar und bearbeitbar.
Mit dieser Funktion kann eine erstellte Referenz wieder entfernt werden.

int RemoveRef (int ObjId, int RefNo)

ObjId : Interne ELO Id
RefNo : Nummer der zu löschenden Referenz (beginnend mit 1)

Rückgabewert:

0	- Referenz gelöscht
-1	- Kein Arbeitsbereich aktiv
-2	- Referenz nicht vorhanden
-3	- löschen fehlgeschlagen
-4	- kein lese/schreibzugriff auf das Objekt
-5	- Nur Zusätzliche Referenzen können gelöscht werden

Bsp.:

Entfernt die fünfte Referenz von Objekt mit der Id 34.
Elo.RemoveRef (34,5)

siehe auch: InsertRef
GetObjRef

Funktion RotateFile

Mit dieser Funktion können Sie eine Bilddatei in der Postbox um 90, 180 oder 270 Grad drehen. In der aktuellen Version lassen sich nur Einseiten-Tiffs drehen, mehrseitige Dokumente können nicht bearbeitet werden.

Als Dateiname kann ein Eintrag aus der Postbox übergeben werden (ohne Pfad, nur der Dateiname), ein Index in die Postliste (durch ein # gekennzeichnet, z.B. #0 ist der erste Eintrag in der Postliste) oder auch ein Leerstring. In diesem Fall wird die aktuelle Postboxdatei (ActivePostFile, z.B. vom vorhergehenden Scanvorgang) verwendet.

int RotateFile(AnsiString FileName, int Degree)

Parameter:

FileName:	Name der zu drehenden Datei
Degree:	Drehwinkel, 90, 180 oder 270

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehlender Dateiname
-3: Fehler beim Lesen der Datei
-4: Datei konnte nicht rotiert oder gespeichert werden
1: ok

Siehe auch: AnalyzePostfile
UpdatePostbox

Funktion RunEloScript

Diese Funktion startet ein ELO-Script. Die Script-Datei wird im Verzeichnis *Postbox\EloScripts* gesucht.

int RunEloScript(AnsiString FileName)

Parameter:

FileName	auszuführendes Script (ohne Extension)

Rückgabewerte:

-1: Script-Datei nicht vorhanden
0: Ok

Siehe auch: DoExecute

Funktion SaveDocumentPage

Mit dieser Funktion kann eine Seite einer Multipasge-TIFF-Datei abgespeichert werden.

int SaveDocumentPage(AnsiString sFileName, int iPage, int iQuality)

Parameter:

sFileName:	Name der abzuspeichernden Datei
iPage:	Seitennummer („1“-indiziert)
iQuality:	Qualität bei JPEG-Bildern (0..100)

Rückgabewerte:

-1: kein aktiver Arbeitsbereich vorhanden
-2: kein Viewer sichtbar
-3: Fehler beim Abspeichern
1: Ok

Siehe auch:

Funktion SaveDocumentZoomed

Diese Funktion erlaubt es, das aktuell angezeigte Dokument skaliert abzuspeichern.

int SaveDocumentZoomed(AnsiString sFileName, int iZoom, int iQuality)

Parameter:

sFileName:	Name der abzuspeichernden Datei
iZoom:	Zoomfaktor in [%]
iQuality:	Qualität bei JPEG-Bildern (0..100)

Rückgabewerte:

-1: kein aktiver Arbeitsbereich vorhanden
-2: kein Viewer sichtbar
-3: Fehler beim Abspeichern
1: Ok

Siehe auch:

Funktion SaveObject

Mit dieser Funktion können Sie die intenen Objekt-Verschlagwortungsdaten zwischenspeichern und wiederherstellen. Diese Funktion darf nicht rekursiv verwendet werden, da es nur einen Sicherungsspeicher gibt. Jeder Save-Vorgang überschreibt den vorhergehenden. Allerdings kann ein Restore mehrfach durchgeführt werden, es werden dann jedesmal die gleichen Daten wiederhergestellt.

int SaveObject(int Mode)

Parameter:

Mode: 1: Sichern, 2: Rücksichern

Rückgabewerte:

-1: Ungültiger Parameter
1: Ok

Siehe auch:

Property ScriptActionKey (int)

Über dieses Property werden Informationen vom Script an ELO zurückgeliefert. ELO reagiert darauf mit ganz bestimmten Aktion.

Programmkontext	**Wert**	**Aktion von ELO**
Aufruf eines Skripts beim Betreten einer Verschlagwortungsmaske	10	Verschlagwortungsmaske wird mit ‚Ok' verlassen
	-10	Verschlagwortungsmaske wird mit ‚Abbrechen' verlassen
Skripte: vor dem Importieren/Exportieren	0	Nicht Importieren/Exportieren
	1	Importieren/Exportieren
Steuerung Eingabefokus in der Verschlagwortungsmaske innerhalb von Skripten, die beim Verlassen oder Betreten von Eingabefeldern ablaufen	11	Feld „Kurzbezeichnung" erhält Eingabefokus
	12	Feld „Memo" erhält Eingabefokus
	13	Feld „Datum" erhält Eingabefokus
	1000+n (n=0..49)	Eingabezeile n erhält Eingabefokus
Verfügbar ab: Ver 3.00.228		
Nach OK Click in der Verschlagwortungsmaske	-20	Abbrechen des OK Vorgangs
Verfügbar ab: Ver 3.00.350		
Haftnotiz	0	Nach dem Bearbeiten einer Haftnotiz
	1	Vor dem Bearbeiten einer Haftnotiz
Beim Eintragen/Verschieben einer Objektrefrenz	-30	Hiermit wird das Verschieben verhindert

siehe auch:

Property SearchListColumns (AnsiString)

Mit diesem Property können Sie die Spaltenüberschriften der Trefferliste ermitteln. Die einzelnen Überschriften sind durch ein „|“-Zeichen getrennt. Der String kann mit Hilfe der VBScript-Funktion „Split“ in einzelne Strings zerlegt werden. Das Property wird benötigt, wenn die Trefferliste mit Hilfe der Funktion „SortSearchList“ sortiert werden soll. Es kann dann sichergestellt werden, dass bei der Sortierung die richtige Spalte angesprochen wird.

siehe auch: SortSearchList

Funktion SearchListLineId

Mit dieser Funktion kann die ELO Objektid einer Zeile aus der Suchliste ermittelt werden.

int SearchListLineId(int LineNo)

Parameter:

LineNo	zu selektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
>0: ObjektId

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )

for i = 0 to 8
  res = res & Elo.SearchListLineSelected( i ) & " - " &
Elo.SearchListLineId( i ) & ", "
next

MsgBox res
```

siehe auch: UnselectSearchListLine
SelectSearchListLine
SearchListLineSelected

Funktion SearchListLineSelected

Mit dieser Funktion kann getestet werden, ob eine Zeile in der Suchansicht selektiert ist.

int SearchListLineSelect ed(int LineNo)

Parameter:

LineNo	zu selektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
0: Zeile nicht selektiert
1: Zeile selektiert

siehe auch: UnselectSearchListLine
SelectSearchListLine

Funktion SelectAllPostBoxLines

Diese Funktion selektiert alle Zeilen der Postbox.

int SelectAllPostBoxLines()

Parameter:

Keine

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: SelectPostBoxLine
UnselectPostBoxLine
PostBoxLineSelected
UnselectAllPostBoxLines

Funktion SelectArcListLine

Diese Funktion selektiert einen Eintrag in der Suchansicht. Die Zeilennummer läuft dabei von 0 bis Anzahl der Einträge minus 1.

int SelectArcListLine(int LineNo)

Parameter:

LineNo	zu selektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
1: Ok

Beispiel:

```
Set Elo = CreateObject( "ELO.office" )

for i = 0 to 8
  res = res & Elo.ArcListLineSelected( i ) & " - "
next

for i = 0 to 3
  Elo.SelectArcListLine( i )
next

for i = 4 to 7
  Elo.UnselectArcListLine( i )
next

Elo.SelectArcListLine( 8 )

MsgBox res
```

siehe auch: UnselectArcListLine
ArcListLineSelected

Funktion SelectLine

Über die Funktion SelectLine können Sie einen Eintrag der aktuellen Auswahlliste selektieren. Hierzu wird in der Archivansicht die linke Seite, ansonsten die Klemmbrettliste, Postliste, Rechercheliste bzw die Wiedervorlageliste verwendet.

int SelectLine(int LineNo)

Parameter:

LineNo: Zu selektierende Zeile

Rückgabewerte:

-3: Kein Arbeitsbereich aktiv
-2: Keine Auswahlliste verfügbar
-4: Ende der Liste erreicht

ansonsten: Ausgewählte Zeile vor der Operation

Siehe auch: SelectView

Funktion SelectPostBoxLine

Funktion SelectPostBoxLineEx

Diese Funktion selektiert einen Eintrag in der Postbox. Mit Hilfe der Funktion SelectPostBoxLineEx kann das Dokument neu in den Viewer geladen werden.

int SelectPostBoxLine(int LineNo)
int SelectPostBoxLineEx(int LineNo, int Mode)

Parameter:

LineNo	zu selektierende Zeile
Mode	1=Dokument wird neu in den Viewer geladen

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: UnselectPostBoxLine
PostBoxLineSelected
SelectAllPostBoxLines
UnselectAllPostBoxLines

Funktion SelectSearchListLine

Diese Funktion selektiert einen Eintrag in der Suchansicht. Die Zeilennummer läuft dabei von 0 bis Anzahl der Einträge minus 1.

Ab der Version 3.00.544 gibt es zusätzlich noch die Möglichkeit, dass Sie eine negative Zeilennummer eingeben. In diesem Fall wird nicht nur die Zeile selektiert sondern es wird zusätzlich noch die Dokumentenansicht auf diese Zeile aktualisiert. Da man für die oberste Zeile nicht zwischen einer 0 und einer "minus 0" unterscheiden könnte, beginnt diese Zeilennummer bei –1 statt bei –0 für den ersten Eintrag.

int SelectSearchListLine(int LineNo)

Parameter:

LineNo	zu selektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: UnselectSearchListLine
SearchListLineSelected

Funktion SelectTreePath

Dieser Befehl aktiviert in der Suchansicht den TreeView und blättert ihn gemäß des angegebenen Parameterstrings auf. Die einzelnen Stufen sind dabei jeweils durch ein ¶-Trennsymbol gekennzeichnet, die letzte Ebene wird selektiert und nicht weiter aufgeblättert. Dabei kann man für die letzte Ebene noch die Platzhalter "*" für "ersten Eintrag selektieren" und "-" für "nichts selektieren" angeben.

int SelectTreePath(AnsiString TreePath)

Parameter:

TreePath	Aufzublätternder Pfad im TreeView

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: Ok

Beispiel:

Das folgende Beispiel führt eine Suche über alle EMails (Maskennummer 2) durch, die im Absender irgendwo "Thiele" stehen haben. Anschließend wird der Pfad Lager - Einkauf - Lagereingang aufgeblättert und dort der erste Untereintrag selektiert:

```
Set Elo=CreateObject("ELO.office")

Elo.SelectView 4
Elo.PrepareObjectEx 0, 0, 2
Elo.SetObjAttrib 0, "%Thiele%"
Elo.DoSearch
Elo.SelectTreePath "Lager¶Einkauf¶Lagereingang¶*"
```

siehe auch: SelectView

Funktion SelectUser

Funktion SelectUserEx

Über die Funktion SelectUser und SelectUserEx können Sie einen Auswahldialog zur Selektion eines Anwenders aufrufen. Sie erhalten eine Liste aller Anwender (optional ohne Ihrem eigenen Eintrag) und erhalten als Rückgabe die ausgewählte AnwenderId (oder –1 bei Abbruch).

int SelectUser(int SuppressOwnName)

Parameter:

SuppressOwnName	0: Alle Anwender, einschließlich des eigenen Namens 1: Eigener Name wird nicht mit zur Auswahl angeboten.

Rückgabewerte:

-1: Kein Anwender ausgewählt
>=0: AnwenderId

AnsiString SelectUserEx(int SelectFlags)

Parameter:

SelectFlags	Wert:	1: MultiSelect in der Anwenderauswahl erlauben 16: Anwendernamen – Nachname vorziehen 32: Gruppennamen – Nachname vorziehen 256: Eigenen Namen in der Liste nicht aufführen

Rückgabewerte:

Leer: Kein Anwender ausgewählt
Sonst: Anwenderliste, Nummern durch Komma getrennt bei Mehrfachauswahl

Beispiel:

```
Set Elo=CreateObject("ELO.office")
Users = Elo.SelectUserEx(1)
UserList = split( Users, "," )
for each UserNo in UserList
  MsgBox Elo.LoadUserName(UserNo)
Next
```

siehe auch: FindUser
LoadUserName
ActiveUserId

Funktion SelectView

Über die Funktion SelectView können Sie bestimmen welche Arbeitsoberfläche sichtbar sein soll. Weiterhin können Sie damit abfragen, welche gerade sichbar ist.

int SelectView(int ViewId)

Parameter:

ViewId	0: Abfrage der aktuellen Ansicht, Rückgabewert 1..5 1: Archivansicht 2: Klemmbrett 3: Postbox 4: Recherche 5: Wiedervorlage 6: Nachrichten 7: (nur Government Version) Akten 8: In Bearbeitung

Rückgabewerte:

-2: Ungültiger Wert für ViewId
-1: Kein Arbeitsbereich aktiv
1: Bei ViewId>0: Neue Ansicht aktiv

siehe auch: SelectLine

Funktion SelectWorkArea

Im Normalfall wirken alle Automation-Befehle, die das UserInterface betreffen (z.B. GotoId) auf die aktuell aktive Ansicht. Wenn Sie mit einer zweiten Ansicht arbeiten, können Sie die beiden Fenster nicht unabhängig voneinander steuern. Über die Funktion SelectWorkArea können Sie alle nachfolgenden Befehle auf eine der beiden Ansichten festlegen. Nach dem Abschluß der Fensterbezogenen Aktione sollten Sie wieder mit SelectWorkArea(0) auf die Standardeinstellung zurücksetzen. Beachten Sie, dass bei längeren Aktionen ein Arbeitsbereich mittlerweile ungültig geworden sein kann, z.B. wenn der Anwender das Fenster zwischenzeitlich geschlossen hat. Aus diesem Grund sollte die explizite Wahl immer nur kurzfristig gesetzt werden und nach Abschluß der Arbeiten sofort zurückgesetzt werden.

Beispiel:

```
set Elo = CreateObject("ELO.office")
if Elo.SelectWorkArea(1)>=0 then
  ‘ hier nun die Befehle für den ersten Arbeitsbereich
  x=Elo.SelectView(3) ’ Auf die Postboxansicht umschalten
end if

if Elo.SelectWorkArea(2)>=0 then
  ‘ hier nun die Befehle für den zweiten Arbeitsbereich
  x=Elo.SelectView(5) ‘Auf die Wiedervorlageansicht umschalten
end if

’Nach der Aktion auf die Standardeinstellung zurücksetzen
x=Elo.SelectWorkArea(0)
```

int SelectWorkArea(int ViewNr)

Parameter:

ViewNr.	Nummer des Arbeitsbereichs 0: verwendet den jeweils aktiven Bereich (Standardeinstellung) 1: verwendet den Bereich 1 2: verwendet den Bereich 2

Rückgabewerte:

-1: Ungültiger Arbeitsbereich ausgewählt
-2: Gewählter Bereich nicht aktiv
0,1,2: zuletzt ausgewählter Bereich

Verfügbar

3.00.264

Funktion SelList

Über die Funktion SelList können Sie ELO dazu veranlassen eine hierarchische Liste anzuzeigen. Als Eingabeparameter geben Sie den Dateiname der Liste. Wenn Sie nur einen Namen ohne Pfad angeben, wird die Datei im Postboxverzeichnis erwartet. Als Rückgabewert erhalten Sie einen Leerstring (Abbruch) oder die ausgewählte Option.

AnsiString SelList(AnsiString)

Rückgabewerte:

Leer: Abbruch, keine Auswahl
Text: Ausgewählte Option

siehe auch:

Funktion SeparateTiffFile

Mit der Funktion SeparateTiffFile können Sie eine Multipage Tiff-Datei in Einzeldateien aufsplitten. Diese können dabei automatisch, über Trennseiten gesteuert, wieder zu Teildokumenten zusammengefasst werden.

Für die Einzeldateien wird als Dateiname der Originalname verwendet. Dieser wird jeweils um ein Unterstrich und der Zeilennummer ergänzt (beginnend mit 0). Aus der Datei XYZ.TIF werden also die Einzeldateien XYZ_0.TIF, XYZ_1.TIF ... , diese werden im gleichen Verzeichnis abgelegt.

Achtung: ELO prüft nicht ab, ob für die neu erzeugten Dateien ein Namenskonflikt vorliegt. Falls es unter einem der automatisch generierten Dateinamen bereits einen Eintrag gibt, wird er ohne Rückfrage oder Warnung überschrieben. Falls nicht sichergestellt ist, dass es nicht zu derartigen Konflikten kommen kann, muss das Script diese Kontrolle vor dem Aufruf selber ausführen.

AnsiString SeparateTiffFile(AnsiString FileName, int ScanProfil, int Trennseite, int Leerseite)

Parameter:

FileName:	Pfad- und Dateiname der aufzusplittenden Datei
ScanProfil:	-1: keine Vorgabe, 0..7: ELO Scan Profil (Definition der Trennseite, Leerseite)
Treenseite:	Zu verwendende Trennseite für die Teildokumenterkennung: 1: Keine Trennseite, alle Seiten werden wieder in ein Zieldokument gepackt 2: Balkentrennseite 3: Leerseite als Trennseite 4: Einzelseiten, kein Teildokumentzusammenfassung
Leerseite:	0: Leerseiten erhalten, 1: Leerseiten verwerfen

Rückgabewerte:

Liste der erzeugten Dateinamen, jeweils durch ein Trennsymbol verbunden.

Beispiel:

```
set Elo = CreateObject("ELO.office")
file=Elo.ActivePostFile
if file <> "" then
  MsgBox Elo.SeparateTiffFile( file, -1, 4, 0 )
end if
```

Funktion SetCookie

Die Funktion SetCookie setzt den Wert eines Cookie-Eintrags. Falls das Cookie noch nicht existiert wird es automatisch erzeugt. Die Funktionen Get/SetCookie sind primär dazu gedacht, daß ELO Scripting Macros dauerhaft Informationen in ELO hinterlegen können. Der Cookie-Speicher wird erst beim Beenden von ELO gelöscht.

Zum setzen eines Cookies wird ein Name und ein Wert übergeben. Der Zugriff auf ein Cookie erfolgt über den Namen (Ident), zurückgegeben wird der zugeordnete Wert. Falls das Cookie unbekannt ist wird ein Leerstring zurückgegeben.

Beachten Sie bitte, daß die Anzahl der Cookies begrenzt ist, jedes Makro oder jedes andere externe Programm sollte nur wenige davon verwenden und bei jedem Aufruf die gleichen wieder verwenden statt immer wieder neue anzulegen.

int SetCookie(AnsiString Ident, AnsiString Value)

Parameter:

Ident Cookie-Bezeichnung, wird bei GetCookie verwendet
Value Cookie-Inhalt

Rückgabewerte:

-1: Fehler beim Anlegen des Cookies (z.B. Speicher voll)
1: ok

Siehe auch: GetCookie

Funktion SetObjAttrib

Diese Funktion schreibt den Wert einer Eingabezeile der aktuellen Dokumentenmaske.

Zu jeder Eingabezeile gehören 3 Werte:

Die „sichtbare“ Bezeichnung, sie wird in den Dokumentenmasken dem Anwender als Feldbezeichnung angezeigt (z.B. Rechunungsnummer). Sie dient als Benennung der Zeile für den Menschen.

Die Ergänzung, sie wird in der Datenbank zur Identifikation der Zeile im Index verwendet (z.B. RENR). Sie dient als Benennung der Zeile für die Maschine.

Der Eingabewert, hier wird die Anwendereingabe gespeichert (z.B. 199807106)

int SetObjAttrib(int AttribNo, AnsiString AttribValue)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
AttribValue	Der neue Eingabewert dieser Zeile, maximal 30 Zeichen

Rückgabewerte:

-1: Ungültige Zeilennummer
-2: Kein Schreibrecht
1: ok

siehe auch: GetObjAttrib
SetObjAttribKey
SetObjAttribName
SetObjAttribFlags
SetObjAttribMin
SetObjAttribMax
SetObjAttribType

Funktion SetObjAttribFlags

Diese Funktion setzte die Flags einer Eingabezeile der aktuellen Dokumentenmaske. Der zu übergebende Wert „Flags“ ist vom Typ Integer, die dort gesetzten Bits haben folgende Bedeutung:

Bit 0	Eintragungen nur mit Stichwortliste erlaubt
Bit 1	* automatisch vor Suchtext einfügen
Bit 2	* automatisch nach Suchtext einfügen
Bit 3	Neue Lasche nach dieser Zeile
Bit 4	Zeile unsichtbar
Bit 5	Zeile schreibgeschützt
Bit 6	Spalte mit hoher Priorität, Text in die Kurzbezeichnung aufnehmen

int SetObjAttribFlags(int AttribNo, int Flags)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
Flags	Integer-Wert mit entsprechend gesetzten Bits (s.o.)

Rückgabewerte:

-1: Ungültige Zeilennummer
-2: Kein Schreibrecht
1: ok

siehe auch: SetObjAttrib
SetObjAttribKey
SetObjAttribName
GetObjAttribFlags
SetObjAttribMin
SetObjAttribMax
SetObjAttribType

Funktion SetObjAttribKey

Diese Funktion ändert die Index-Bezeichnung (Gruppenname) einer Zeile der aktuellen Dokumentenmaske.

int SetObjAttribKey(int AttribNo, AnsiString KeyValue)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
KeyValue	Der neue Index dieser Zeile.

Rückgabewerte:

-1: Ungültige Zeilennummer
-2: Kein Schreibrecht
1: ok

siehe auch: SetObjAttrib
GetObjAttribKey
SetObjAttribName
SetObjAttribFlags
SetObjAttribMin
SetObjAttribMax
SetObjAttribType

Funktion SetObjAttribMax

Diese Funktion schreibt die maximale Eingabelänge einer Eingabezeile der aktuellen Dokumentenmaske fest. Dieses Feld wird in der Standard-Eingabemaske innerhalb von ELO ausgewertet. Falls Sie die Eingaben über OLE Automation programmgesteuert vornehmen, müssen Sie selber auf die Einhaltung der minimalen und maximalen Längen einer Eingabe achten.

int SetObjAttribMax(int AttribNo, int AttribMax)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
AttribMax	Maximale Länge der Eingabe, 0: keine Kontrolle

Rückgabewerte:

-1: Ungültige Eingabezeile
-2: Kein Schreibrecht
1: ok

siehe auch: SetObjAttrib
SetObjAttribKey
SetObjAttribName
SetObjAttribFlags
SetObjAttribMin
GetObjAttribMax
SetObjAttribType

Funktion SetObjAttribMin

Diese Funktion schreibt die minimale Eingabelänge einer Eingabezeile der aktuellen Dokumentenmaske fest. Dieses Feld wird in der Standard-Eingabemaske innerhalb von ELO ausgewertet. Falls Sie die Eingaben über OLE Automation programmgesteuert vornehmen, müssen Sie selber auf die Einhaltung der minimalen und maximalen Längen einer Eingabe achten.

int SetObjAttribMin(int AttribNo, int AttribMin)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
AttribMin	Minimale Länge der Eingabe, 0: keine Kontrolle

Rückgabewerte:

-1: Ungültige Eingabezeile
-2: Kein Schreibrecht
1: ok

siehe auch:
SetObjAttrib
SetObjAttribKey
SetObjAttribName
SetObjAttribFlags
GetObjAttribMin
SetObjAttribMax
SetObjAttribType

Funktion SetObjAttribName

Diese Funktion ändert die für den Anwender sichtbare Bezeichnung einer Zeile der aktuellen Dokumentenmaske.

Zu jeder Eingabezeile gehören 3 Werte:

Die „sichtbare“ Bezeichnung, sie wird in den Dokumentenmasken dem Anwender als Feldbezeichnung angezeigt (z.B. Rechunungsnummer). Sie dient als Benennung der Zeile für den Menschen.

Die Ergänzung, sie wird in der Datenbank zur Identifikation der Zeile im Index verwendet (z.B. RENR). Sie dient als Benennung der Zeile für die Maschine.

Der Eingabewert, hier wird die Anwendereingabe gespeichert (z.B. 199807106)

int SetObjAttribName(int AttribNo, AnsiString NameValue)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
NameValue	Die neue Bezeichnung dieser Zeile.

Rückgabewerte:

-1: Ungültige Zeilennummer
-2: Kein Schreibrecht
1: ok

siehe auch: SetObjAttrib
SetObjAttribKey
GetObjAttribName
SetObjAttribFlags
SetObjAttribMin
SetObjAttribMax
SetObjAttribType

Funktion SetObjAttribType

Diese Funktion schreibt die Art der Eingabe einer Eingabezeile der aktuellen Dokumentenmaske fest. Dieses Feld wird in der Standard-Eingabemaske innerhalb von ELO ausgewertet. Falls Sie die Eingaben über OLE Automation programmgesteuert vornehmen, müssen Sie selber auf die Einhaltung der Art einer Eingabe achten.

0: beliebiges Textfeld
1: Datumsfeld
2: Numerisches Feld
3: ISO-Datum
4: Listeneintrag
5: Anwenderfeld
6: Thesaurus
7: Numerisch mit fester Breite
8: Numerisch mit fester Breite, 1 Nachkommastelle
9: Numerisch mit fester Breite, 2 Nachkommastelle
10: Numerisch mit fester Breite, 4 Nachkommastelle
11: Numerisch mit fester Breite, 6 Nachkommastelle
12: Aktenzeichen

int SetObjAttribType(int AttribNo, int AttribType)

Parameter:

AttribNo	Die Zeilen der Eingabemasken werden von 0..49 durchnumerieret. Zeile 51 enthält den Dateinamen des Dokumentes
AttribMax	Zulässiger Typ der Eingabe

Rückgabewerte:

-1: Ungültige Eingabezeile
-2: Kein Schreibrecht
1: ok

siehe auch: SetObjAttrib
SetObjAttribKey
SetObjAttribName
SetObjAttribFlags
SetObjAttribMin
SetObjAttribMax
GetObjAttribType

Funktion SetOption

Mit der Funktion SetOption können Sie einzelne Einstellungen des Optionen-Dialogs temporär ändern. Der Aufruf liefert als Rückgabewert dabei den Wert der aktuellen Einstellung zurück.

AnsiString SetOption(int OptionNo, AnsiString NewValue)

Parameter:

OptionNo:
0: Postbox – Automatische Ablage bei vollständigen Index
1: Name des Fax-Druckers (falls konfiguriert)
2: Name des Tiff-Printers (falls konfiguriert)
3: OM: Name der Outlook Maske
4: OM: Nummer der Indexzeile "Absender"
5: OM: Nummer der Indexzeile "Empfänger"
6: OM: Nummer der Indexzeile "Id"
7: OM: Art der Ablage (0=MSG Format)
8: Client läuft in einer NT-Umgebung
9: Länge einer Indexzeile
10: Länge der Kurzbezeichnung
11: Länge des Memotextes
12: Voreingestellte Postbox Maske
13: Maske für neue Strukturelemente
14: Maske für neue Dokumente
15: Maske für neue Office Dokumente
16: Schwellwert für „großes Dokument“ Warnung
17: Letzte Object-Id Nummer im Archiv (ReadOnly)
18: Dublettenkontrolle
    0: normale Dublettenkontrolle
    1: keine Dublettenkontrolle, immer Eintragen
    2: erste Fundstelle als Referenz eintragen
19: Interner Standardpfad

Rückgabewerte:

Alter Wert der Option oder „ERROR“

Beispiel

```
Set Elo=CreateObject( "ELO.office" )

OldVal=Elo.SetOption( 0, "FALSE" )
MsgBox "Der alte Wert der Option stand auf :" & OldVal & ", er wurde auf
False geändert."

NewVal=Elo.QueryOption(0)
MsgBox "Der neue Wert der Option steht auf : " &NewVal

call Elo.SetOption(0, OldVal)
MsgBox "Der Wert wurde wieder auf den Originalzustand zurückgesetzt."
```

siehe auch: QueryOption

Funktion SetScriptButton

Mit dieser Funktion kann ein Skriptbutton des ELO Hauptbildschirms mit einem Skript belegt werden. Die Belegung wird benutzerabhängig gespeichert.
Die Funktion wird innerhalb von Installationsskripten aufgerufen, mit deren Hilfe Skripte importiert werden können bei gleichzeitiger Einrichtung der Button- und Menübelegung.

int SetScriptButton(int iTabSheet, int iButton, AnsiString sScriptName, int iVisible)

Parameter:

iTabSheet	Auswahl der Ansicht: 1: Archivansicht (bis zu 16 Buttons) 2: Klemmbrett (bis zu 8 Buttons) 3: Postbox (bis zu 8 Buttons) 4: Recherche (bis zu 8 Buttons) 5: Wiedervorlage (bis zu 8 Buttons)
iButton	Nummer des Buttons (1..16 bzw. 1..8)
sScriptName	Name des Skripts (ohne Extension)
iVisible	1 = Button sichtbar schalten 0 = Button unsichtbar schalten

Rückgabewerte:

-4: Skript nicht verfügbar
-3: Ungültiger Wert für iButton
-2: Ungültiger Wert für Archivansicht
-1: Kein Arbeitsbereich aktiv
1: ok

siehe auch: SetScriptMenu
GetScriptButton
ImportScript

Funktion SetScriptEvent

Diese Funktion setzt ein Script-Event.

SetScriptEvent (AnsiString Event, AnsiString Script,int Mode)

Event : String mit dem Event-Bezeichner (siehe Event-Liste)
Oder vorangestellt mit # die Position (diese Werte können in kommenden ELO Versionen variieren.

Script : String mit dem Scriptnamen ohne Endung (wenn im Standart Script Pfad) oder Komplette Pfadangabe mit Dateiname + Endung (Script wird dann automatisch ins Scriptverzeichnis kopiert)
Wird ein leerer String übergeben so wird das gesetzte Event gelöscht. Hier hat Mode keine Bedeutung.

Mode : 0 – Scriptbelegung nur bei freiem Platz ändern (wenn Script einen Pfadnamen Enthält, wird beim umkopieren eine vorhandene Datei nicht überschrieben und das Script wird nicht zugewiesen)
1 – Scriptbelegung immer ändern (wenn Script einen Pfadnamen enthält, wird beim Umkopieren eine vorhandene Datei überschrieben)

Rückgabewerte :

-5 - Fehler beim Kopieren des Scripts
-4 - Datei existiert nicht
-3 - Script Position belegt
-2 - Unbekannter Event
-1 - Modus nicht implementiert
>=0 - Event-Nummer die gesetzt wurde

Bsp.:

Setzt ein Script aus dem Scriptverzeichnis in das OnTimer Event, wenn der Platz unbelegt ist.
ELO. SetScriptEvent (“sEonTimer“, “Mein Timer Script“,0)

Setzt ein Script von Laufwerk A: in das OnTimer Event egal ob es belegt ist.
ELO. SetScriptEvent (“sEonTimer“, “a:\egalScript.VBS“,1)

Setzt ein Script aus dem Scriptverzeichnis in das Event an Position 6
ELO. SetScriptEvent (“#6“, “Mach was Script“, 0)

Löscht ein Script aus Event 12
ELO. SetScriptEvent (“#12“, ““, 0)

Eventliste

Event	Position	Beschreibung
sEonTimer	0	Beim Timer-Aufruf
sEbeforeCollectPBox	1	Vor dem Sammeln des Postboxeinträge
sEafterCollectPBox	2	Nach dem Sammeln der Postboxeinträge
sEbeforeCollectWv	3	Vor dem Sammeln der Wiedervorlage
sEafterCollectWv	4	Nach dem Sammeln der Wiedervorlage

sEafterScan	5	Nach dem Scannen
sEafterBarcode	6	Nach der Barcodeerkennung
sEafterOCR	7	Nach der OCR
sEafterSearch	8	Nach Recherche
sEafterChangeArcDepth	9	Nach dem Wechsel der Archivebene
sEonReworkBarcode	10	Nachbearbeitung Barcode
sEafterMovePBox	11	Nach dem Verschieben aus der Postbox
sEonClickEntry	12	Beim Klicken auf einen Eintrag
sEonWorkWv	13	Beim Bearbeiten der Verschlagwortung
sEbeforeMovePBox	14	Vor dem Verschieben aus der Postbox
sEonEnterArc	15	Beim Betreten des Archivs
sEonLeaveArc	16	Beim Verlassen des Archivs
sEonLoadSaveUser	17	Beim Lesen oder Speichern eines Anwenders
sEafterInsertNewDoc	18	Nach dem Neueintrag eines Dokuments in das Archiv
sEafterSaveWv	19	Nach dem Speichern eines Wiedervorlagetermins
sEbeforeDeleteWv	20	Vor dem Löschen eines Wiedervorlagetermins
sEbeforeExImportEntry	21	Vor dem Exportieren / Importieren eines Eintrags
sEbeforeShowDoc	22	Vor der Anzeige eines Dokuments
sEonCheckInOutDoc	23	Beim Aus-/Einchecken eines Dokuments

sEeditNote	24	Beim Bearbeiten einer Haftnotiz
sEinsertRef	25	Beim Einfügen/ Verschieben einer Referenz
sECollectList	26	Vor dem Sammeln der Nachfolgerliste
sEreadwriteActivity	27	Beim Lesen oder Schreiben einer Aktivität
sEviewerExport	28	Beim Export in den Viewer
sEbeforeSearch	29	Vor der Suche
sEbatchStore	30	Bei der Stapelablage
sEhtmlView	31	Vor der Anzeige der HTML Verschlagwortung

siehe auch: GetScriptEvent

Funktion SetScriptLock

Diese Funktion sperrt die Ausführung weiterer Script-Events. Damit können Sie weitere Events unterdrücken, die evtl. durch Ihre Script-Funktionen aufgerufen werden. Wenn Sie z.B. ein Script Event beim CheckIn haben und dort eigene CheckIn Aktionen ausführen, würden diese ja wiederum Ihr Script aktivieren. Um solche Rekursionen zu vermeiden, wurde die ScriptLock Funktion eingerichtet. Während die Sperre aktiv ist, werden keinen weiteren Scripte aufgerufen.

ACHTUNG: Diese Funktion sollten Sie nur mit großer Vorsicht und nur für sorgfältig ausgewählte Funktionen verwenden. Wenn Sie die Sperre stehen lassen, werden nach Ihrem Script überhaupt keine weiteren Scripte mehr ausgeführt!

int SetScriptLock(int Lock)

Parameter:

Lock	1: Sperre setzen, 0: keine Sperre

Rückgabewerte:

Alter Wert der Sperre

Funktion SetScriptMenu

Mit dieser Funktion kann ein ELO Skript in ein Kontextmenü des ELO Hauptbildschirms eingetragen werden. Jede Ansicht (Archivansicht, Klemmbrett, Postbox, Recherche, Wiedervorlage) besitzt ein eigenes Kontextmenü. Die Belegung wird benutzerabhängig gespeichert.
Die Funktion wird innerhalb von Installationsskripten aufgerufen, mit deren Hilfe Skripte importiert werden können bei gleichzeitiger Einrichtung der Button- und Menübelegung.

int SetScriptMenu(int iTabSheet, AnsiString sScriptName)

Parameter:

iTabSheet	Auswahl der Ansicht: 1: Archivansicht 2: Klemmbrett 3: Postbox 4: Recherche 5: Wiedervorlage
sScriptName	Name des Skripts (ohne Extension)

Rückgabewerte:

-2: Ungültiger Wert für Archivansicht
-1: Kein Arbeitsbereich aktiv
1: ok

siehe auch: SetScriptButton
GetScriptButton
ImportScript

Funktion ShowAutoDlg ()

Verfügbar ab:	**Ver 3.00.228**

Zeigt den erstellten Dialog. **Es muss vorher ein CreateAutoDlg aufgerufen werden.** Ein mehrmaliger Aufruf von ShowAutoDlg hintereinander ist nicht möglich, denn nach OK oder Abbruch werden die Informationen, um den Dialog anzeigen zu können, zerstört. Es ist also ein erneuter Aufruf von CreateAutoDlg erforderlich.

int ShowAutoDlg ()

Rückgabewert :

0 – Abbrechen wurde gedrückt.
1 – OK wurde gedrückt.

Bsp.:

Erstellt einen leeren Dialog und zeigt ihn an.
Elo.CreateAutoDlg („Mein Dialog“)
Elo.ShowAutoDlg

siehe auch: CreateAutoDlg
AddAutoDlgControl
GetAutoDlgValue

Funktion ShowDocInverted

Diese Funktion stellt das aktuell angezeigte TIFF-Dokument invertiert dar.

int ShowDocInverted()

Parameter:

keine

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Kein aktiver Viewer sichtbar
1: ok

siehe auch:

Property SigId (int)

Das Property SigId bestimmt die Dokumentenmanager ID der Signatur eines Dokuments. Ein falscher Eintrag an dieser Stelle führt zur Zerstörung der Signatur.

siehe auch: MaskKey
DocKey
DocKind
DocPath

Funktion Sleep

Diese Funktion führt eine Pause mit einer einstellbaren Anzahl von Millisekunden aus. Dabei können Sie zusätzlich vor und/oder nach der Wartezeit ein ProcessMessages ausführen. Dieser Aufruf führt dazu, dass der Client aktiv bedienbar bleibt.

int Sleep(int Flags, int Delay)

Parameter:

Flags	Bit 0: ProcessMessages vor dem Sleep Bit 1: ProcessMessages nach dem Sleep Rest: reserviert
Delay	Wartezeit in Millisekunden

Rückgabewerte:

1: ok

siehe auch:

Funktion SplitFileName

Die Funktion SplitFileName ermittelt aus einem Dateipfad (Laufwerk – Pfad – Dateiname bzw. Server – Pfad – Dateiname) den Pfadanteil mit Laufwerk oder Server, den Dateiname oder die Dateikennung

AnsiString SplitFileName(AnsiString FilePath, int iMode)

Parameter:

FilePath	Vollständiger Dateiname mit Laufwerk und Pfad
iMode	1: Gibt den Laufwerk+Pfadanteil zurück 2: Gibt den Dateinamen zurück 3: Gibt die Dateikennung zurück

Funktion StartScan

Über die Funktion StartScan können Sie einen Scanvorgang in der Postbox auslösen. Über den Parameter ActionCode können Sie vorgeben, ob Einzelseiten eingescannt werden oder ob ein Stapelscan ausgeführt wird. Der zweite Parameter bestimmt, welche Scannervoreinstellungen verwendet werden (Seitengröße etc.).

Das eingelesene Image können Sie über ActivePostFile ermitteln. Wenn dieser Eintrag leer ist, hat es beim Scannen einen Fehler gegeben oder der Anwender hat den Vorgang abgebrochen.

int StartScan (int ActionCode, int PageSize)

Parameter:

ActionCode	0: kein Scanlauf, nur die Seitengröße setzen 1: Stapelscannen 2: Einzelseitenscan
PageSize	0: mit Vorrausschau scannen 1..4: mit Scannerparametersatz 1..4 (Optionen – Scanner)

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: ok

siehe auch:

Funktion Status

Diese Funktion setzt im aktuellen ELO Hauptfenster einen Text in die Statuszeile.

int Status(AnsiString Text)

Parameter:

Text: auszugebender Text

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: ok

Siehe auch: UpdatePostbox

Funktion StoreDirect

Diese Funktion legt die in der Postbox selektierten Einträge im Archiv ab, und entspricht damit dem Menüpunkt ‚Direktablage ins Archiv' im Kontextmenü der Postbox. Vor dem Aufruf muß in der Archivansicht das gewünschte Register geöffnet werden.

int StoreDirect ()

Parameter:

keine

Rückgabewerte:

-3: kein Register selektiert
-2: keine Einträge in der Postbox selektiert
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: StoreKeyword
StoreMulti

Funktion StoreKeyword

Diese Funktion legt die in der Postbox selektierten Einträge per Schlagwortablege im Archiv ab, sie entspricht dem Menüpunkt ‚Schlagwortablage ins Archiv' im Kontextmenü der Postbox.

int StoreKeyword ()

Parameter:

keine

Rückgabewerte:

-2: keine Einträge in der Postbox selektiert
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: StoreDirect
StoreMulti

Funktion StoreMulti

Diese Funktion legt die in der Postbox selektierten Einträge im Archiv ab, sie entspricht dem Menüpunkt ‚Anonyme Registerablage ‘ im Kontextmenü der Postbox.

int StoreMulti()

Parameter:

keine

Rückgabewerte:

-3: kein Register selektiert
-2: keine Einträge in der Postbox selektiert
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: StoreDirect
StoreKeyword

Property TextParam (AnsiString)

Das Property TextParam enthält unterschiedliche Parameter, die von einem Script-Event an das Script oder von dem Script zurück an das Event übergeben werden sollen. Der Inhalt dieses Properties wird beim jeweiligen Script Event beschrieben. Zu allen anderen Zeitpunkten hat dieses Property einen zufälligen Wert und sollte auch nicht verändert werden.

Funktion ToClipboard

Diese Funktion übergibt einen Text an das Windows Clipboard

Int ToClipboard(AnsiString Text)

Parameter:

Text Zu übergebender Text

Rückgabewerte:

Immer o

Beispiel:

```
Option Explicit

Dim Elo,url,id,Guid,iObjID,iZeile,Body

set Elo=CreateObject("ELO.office")
url="http://hobbit3:8081/guid/"

if Elo.SelectView(0)=1 then
  Id=Elo.GetEntryId(-1) ' Selektierten Eintrag wählen
  if Id>1 then
    if Elo.PrepareObjectEx(Id,0,0)>0 then
      Guid=Mid(Elo.ObjGuid,2, Len(Elo.ObjGuid)-2)
      Elo.ToClipboard url & Guid
    end if
  end if
elseif Elo.SelectView(0)=2 or Elo.SelectView(0)=4 then
  iZeile=0
  Body=""
  Do
    iObjID=Elo.GetEntryID(iZeile)
    If iObjID=0 Then Exit Do
    Elo.PrepareObjectEx iObjID,0,0
    If Elo.ObjTypeEx=254 Then
      Guid=Mid(Elo.ObjGuid,2, Len(Elo.ObjGuid)-2)
      Body=Body & url & Guid & vbCrLf & vbCrLf
    End If
    iZeile=iZeile+1
  Loop Until False
  Elo.ToClipboard Body
end if
```

Siehe auch: FromClipboard

Funktion TreeWalk

Diese Funktion läuft vom Startknoten über alle Unterknoten und ruft für jeden Eintrag ein Skript auf. Hiermit können Sie also ganze Strukturen durch einen Aufruf bearbeiten lassen.

Beim Abarbeiten der Unterobjekte haben Sie über den Mode Parameter die Option, den aktuellen Knoten vor den Unterknoten oder nach den Unterknoten (oder vor und nach) zu Bearbeiten. Wenn Sie ein Attribut eines Schrankes an alle Dokumente durchreichen wollen, dann ist der Aufruf "vor" notwendig (das Ordnerattribut muss gesetzt werden, bevor die Register bearbeitet werden). Wenn Sie Informationen (z.B. Anzahl der Dokumente, die einem bestimmten Kriterium entsprechen) einsammeln wollen, dann ist ein "nach" angemessen (es müssen erst alle Register gezählt werden, damit die Ordnersumme ermitteln kann. Falls die "Knoten-Nachfolger"-Reihenfolge unwichtig ist (z.B. "drucke alle Dokumente"), dann sollten Sie die "vor" Methode wählen, da sie etwas weniger Systemlast verursacht.

Über den Mode Parameter können Sie zudem noch bestimmen, ob Sie auch Referenzen oder nur Hauptvorgängerwege durchlaufen wollen.

Über den Parameter MaxDepth können Sie maximale Bearbeitungstiefe begrenzen. Wenn Sie z.B. alle Ordner bearbeiten wollen, dann setzen Sie den Startknoten auf 1 (Archivobjekt) und die Maximale Tiefe auf 2 (Schrank- und Ordnerebene). Im Callback-Skript müssen Sie dann nur noch nachprüfen, ob der Aufruf zu einem Schrankobjekt erfolgt (dann ist nichts zu tun) oder zu einem Ordnerobjekt. Die Suche geht von hier aus aber nicht weiter in die Tiefe, so dass dieser Aufruf sehr effizient abgearbeitet werden kann. Falls Sie alle Objekte aufgelistet bekommen wollen, so können Sie die Maximale Tiefe einfach auf 15 oder eine andere hinreichend große Zahl setzen.

Diese Funktion soll einen einfachen Weg bieten, ganze Hierarchien zu bearbeiten. Sie ist nicht auf maximale Performance ausgelegt, da ja für jeden Eintrag ein eigener Skriptaufruf stattfinden muss. Falls es also um extrem große Datenmengen oder um einen schnellstmögliche Zugriff geht, dann müssen Sie den TreeWalk im Skript selber erstellen. In vielen Fällen dürfte diese Version aber schnell genug sein.

int TreeWalk(int Mode, int StartObj, int MaxDepth, AnsiString ScriptName)

Parameter:

Mode	Bit 0	Der Parentknoten wird vor den Childknoten bearbeitet
	Bit 1	Der Parentknoten wird nach dem Childknoten bearbeitet
	Bit 2	Es werden keine Referenzen beachtet
StartObj		ELO Objektnummer des Startknotens
MaxDepth		Maximale Suchtiefe
ScriptName		Name des Callback Skripts welches für jedes Objekt aufgerufen wird.

Rückgabewerte:

-2: Startknoten nicht gefunden
-1: Kein Arbeitsbereich aktiv
>0: Anzahl der gefundenen Objekte

Beispiel, Skript 1 mit dem Namen **"Callback"**

```
set Elo = CreateObject("ELO.office")
Action=Elo.GetCookie( "ELO_WALK" )
Action=Action & (15-Elo.ActionKey) & ": " & Elo.ObjShort & vbcrlf
call Elo.SetCookie( "ELO_WALK", Action )
```

Skript 2, welches den eigentlichen TreeWalk Aufruf enthält und eine Liste aller Kurzbezeichnungen unterhalb des aktuell selektierten Knotens enthält.

```
set Elo = CreateObject("ELO.office")
call Elo.SetCookie( "ELO_WALK", "" )
cnt=Elo.TreeWalk( 1, Elo.GetEntryId(-1), 15, "callback" )
Action=Elo.GetCookie( "ELO_WALK" )
Action="TreeWalk: "&cnt&" Entries."&vbcrlf&vbcrlf&Action
MsgBox Action
```

siehe auch:

Funktion UnselectAllPostBoxLines

Diese Funktion deselektiert alle Zeilen der Postbox.

int UnselectAllPostBoxLines()

Parameter:

Keine

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: SelectPostBoxLine
UnselectPostBoxLine
PostBoxLineSelected
UnselectAllPostBoxLines

Funktion UnselectArcListLine

Diese Funktion deselektiert einen Eintrag in der rechten Liste der Archivansicht.

int UnselectArcListLine(int LineNo)

Parameter:

LineNo	zu deselektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
1: Ok

Verfügbar seit:
5.00.036

Beispiel

```
Set Elo = CreateObject( "ELO.office" )

for i = 0 to 8
  res = res & Elo.ArcListLineSelected( i ) & " - "
next

for i = 0 to 3
  Elo.SelectArcListLine( i )
next

for i = 4 to 7
  Elo.UnselectArcListLine( i )
next

Elo.SelectArcListLine( 8 )

MsgBox res
```

siehe auch: SelectArcListLine
ArcListLineSelected

Funktion UnselectLine

Diese Funktion deselektiert einen Eintrag in der aktuellen Ansicht. Diese Funktion darf nur in Ansichten angewendet werden, in welchen Mehrfachselektionen (z.B. Klemmbrett) zulässig sind. In den anderen Ansichten (z.B. Archivansicht) kann es zu zufälligen Ergebnissen kommen.

int UnselectLine(int LineNo)

Parameter:

LineNo	zu deselektierende Zeile

Rückgabewerte:

-4: ungültige Zeilennummer
-3: Kein Arbeitsbereich aktiv
-2: Ungültige Zeilennummer
1: Ok, war selektiert
2: Ok, war nicht selektiert

siehe auch: SelectPostBoxLine
PostBoxLineSelected
SelectAllPostBoxLines
UnselectAllPostBoxLines

Funktion UnselectPostBoxLine

Diese Funktion deselektiert einen Eintrag in der Postbox.

int UnselectPostBoxLine(int LineNo)

Parameter:

LineNo	zu deselektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: SelectPostBoxLine
PostBoxLineSelected
SelectAllPostBoxLines
UnselectAllPostBoxLines

Funktion UnselectSearchListLine

Diese Funktion deselektiert einen Eintrag in der Suchansicht.

int UnselectSearchListLine(int LineNo)

Parameter:

LineNo	zu deselektierende Zeile

Rückgabewerte:

-2: ungültige Zeilennummer
-1: Kein Arbeitsbereich aktiv
1: Ok

siehe auch: SelectSearchListLine
SearchListLineSelected

Funktion UpdateDocument, UpdateDocumentEx

Diese Funktion fügt an ein ELO Dokument eine neue Imagedatei an. Je nach Status des Dokuments wird die alte Imagedatei überschrieben (frei Bearbeitung), eine neue Version angelegt (Versionskontrolliert) oder die Funktion zurückgewiesen (Revisionssicher).

int UpdateDocument(int DocId, int Status, AnsiString DocFilePath)
int UpdateDocumentEx(int Id, int Status, AnsiString FilePath, AnsiString Version, AnsiString Comment)

Parameter:

DocId	Interne ELO ObjektId an das die neue Image Datei angefügt werden soll
Status	1: Zugriffsrecht Ablagepfad prüfen, 2: Dokumente bearbeiten Recht prüfen.
DocFilePath	Pfad und Name der neuen Imagedatei

Zusätzlich bei UpdateDocumentEx:

Version	Versionsinfo, freier Text, bis zu 10 Zeichen lang
Comment	Kommentar zur neuen Version, freier Text

Rückgabewerte:

-1: Kein aktiver Arbeitsbereich
-2: Schreib-Zugriff auf das Dokument verweigert
-3: DocId zeigt nicht auf ein Dokument
-4: Fehler beim Lesen der Dokumentendaten aus der Datenbank
-5: Fehler beim Einfügen der Imagedatei in das Archiv
-6: Fehler beim Eintrages des Images in die Datenbank
1: ok

siehe auch: InsertDocAttachment
GetDocumentPath

Funktion UpdateObject

Über die Funktion UpdateObject fügen Sie einen Eintrag in die Datenbank ein. Dieses Objekt muß vorher mittels PrepareObject erzeugt bzw. aus der Datenbank gelesen worden sein. Falls es sich um ein neues Objekt handelt, wird vor dem Speichern nachgeprüft, ob über IndexText ein zulässiges Vorgängerobjekt identifiziert werden kann.

int UpdateObject()

Parameter:

keine

Rückgabewerte:

-4: Fehler beim Update in der Datenbank
-3: Fehlendes oder unbekanntes Ablageziel
-2: Fehler beim Neueintrag in die Datenbank
-1: Kein Arbeitsbereich aktiv
1: Neueintrag vorgenommen
3: Update vorgenommen

siehe auch: PrepareObject
LookupIndex

Funktion UpdatePostbox

Die Funktion UpdatePostbox löst ein erneutes Sammeln des Postbox-Inhaltes aus. Neben den reinen Postbox-Dateien werden noch die Tiff-Printer Dateien, Netzwerkscanner-Dateien und Outlook-Einträge übernommen.

int UpdatePostbox()

Parameter:

keine

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
>=0: Anzahl der Einträge in der Postbox

siehe auch: UpdateObject
UpdateDocument
UpdatePostboxEx

Funktion UpdatePostboxEx

Über die Funktion UpdatePostboxEx lösen Sie entweder ein erneutes Sammeln des Postbox-Inhaltes aus oder Sie selektieren gezielt einen Eintrag aus der Postbox. Beim Sammeln werden neben den reinen Postbox-Dateien noch die Tiff-Printer Dateien, Netzwerkscanner-Dateien und Outlook-Einträge übernommen.

int UpdatePostboxEx(int iMode, int iLine)

Parameter:

iMode: 0: Sammeln der Postbox, wie bei UpdatePostbox
1: Selektieren des aktuellen PB-Eintrags durch Klick auf das Icon
2: Selektieren des aktuellen PB-Eintrags durch Klich auf die Textzeile
3: Selektieren der mitgegebenen Zeilennummer durch Klick auf das Icon
4: Selektieren der mitgegebenen Zeilennummer durch Klick auf den Text
5: Freigabe des aktuellen Postboxviewers

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
>=0: Anzahl der Einträge in der Postbox (bei Sammeln) oder 1

siehe auch: UpdateObject
UpdateDocument
UpdatePostbox

Funktion UpdateSw

Diese Funktion ändert ein Stichwort in einer Stichwortliste. Jeder Eintrag einer Stichwortliste wird innerhalb seiner Ebene eindeutig durch eine 2-stellige Buchstabenkombination AA, AB, AC .. ZZ gekennzeichnet. Durch die Verkettung aller Kennzeichen aller Ebenen können Sie jedes Stichwort im Baum eindeutig ansprechen. Beginnend mit dem Punkt für die Wurzel können Sie nun z.B. über „.AAABAC“ in der untersten Ebene den ersten Eintrag (AA), darunter den zweiten (AB) und darunter den dritten (AC) Eintrag ansprechen

Die Gruppe kennzeichnet die Listenzuordnung zu einer Indexzeile (gleicher Eintrag wie im Gruppenfeld des Maskeneditors).

int UpdateSw(AnsiString Gruppe, AnsiString Parent, AnsiString Wort)

Parameter:

Gruppe	Wählt die Stichwortliste aus
Parent	Vorgängerknoten-Pfad für den geänderten Eintrag
Wort	Neues Stichwort

Rückgabewerte:

-1: kein Workspace offen
-2: Fehler beim Speichern des Stichwortes
1t: Ok

Beispiel

```
...
Set Elo=CreateObject("ELO.office")

call Elo.DeleteSwl( "THM", "." )

MsgBox Elo.AddSw( "THM", ".", "1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.1" )
MsgBox Elo.AddSw( "THM", ".AA", "1.2" )
MsgBox Elo.AddSw( "THM", ".AA", "1.3" )
MsgBox Elo.AddSw( "THM", ".", "2" )
MsgBox Elo.AddSw( "THM", ".", "3" )

MsgBox Elo.ReadSwl( "THM", ".", " - " )
MsgBox Elo.ReadSwl( "THM", ".AA", " - " )

call Elo.UpdateSw( "THM", ".AB", "2a" )
MsgBox Elo.ReadSwl( "THM", ".", " - " )
...
```

Property UserFlags (int)

Das Property UserFlags enthält die Rechte des Anwenders (als Bitmaske). Ein Administrator kann dabei beliebige Rechte zuteilen, ein Subadministrator maximal seine eigenen Rechte. Falls er mehr zuweist, als er selber besitzt, führt das nicht zu einer Fehlermeldung, die Einstellung wird automatisch angepaßt.

```
// Konstanten für Rechte
#define LUR_ADMINISTRATOR     1            Hauptadministrator
#define LUR_EDITSYSTEM        2            Stammdaten bearbeiten
#define LUR_EDITARC           4            Archive bearbeiten
#define LUR_EDITDOCS          8            Dokumente bearbeiten
#define LUR_CHANGEPWD         16           Paßwort ändern
#define LUR_CHANGEREVISION    32           Revisionslevel ändern
#define LUR_EDITANWENDER      64           Anwenderdaten bearbeiten
#define LUR_WFADMINISTRATE    128          Workflows verwalten
#define LUR_WFSTART           0x100        Workflows starten
#define LUR_DELDOCS           0x200        Dokumente löschen
#define LUR_DELSORD           0x400        Aktenstrukturelemente löschen
#define LUR_ARCHIVAR          0x800        !!! Findet sich nicht in der Rechteliste von ELO
                                           !!!
#define LUR_SAPADMIN          0x1000       SAP-Administrator
#define LUR_IMPORT            0x2000       Importberechtigung
#define LUR_EXPORT            0x4000       Exportberechtigung
#define LUR_EDITMASK          0x8000       Ablagemasken bearbeiten
#define LUR_EDITSCRIPT        0x10000      Scripte bearbeiten
#define LUR_EDITDELDATE       0x20000      Verfallsdatum bearbeiten
#define LUR_EDITSWL           0x40000      Stichwortlisten bearbeiten
#define LUR_DELETEREADONLY    0x80000      Revisionssichere Dokumente löschen
#define LUR_EDITREPL          0x100000     Replikationskreise bearbeiten
#define LUR_CHANGEACL         0x200000     Berechtigungseinstellungen verändern
#define LUR_IGNOREACL         0x400000     Alle Einträge sehen, Berechtigungseinstellungen
                                           missachten
#define LUR_EDITSCAN          0x800000     Scannereinstellungen und Profile verändern
#define LUR_CHANGEMASK        0x1000000    Maskentyp nachträglich verändern
#define LUR_ACTPROJ           0x2000000    Projekte für Aktivitäten einrichten
#define LUR_CHANGEPATH        0x4000000    Ablagepfadeinstellung verändern
#define LUR_NOLOGIN           0x8000000    Anmeldesperre aktivieren
#define LUR_DELVERSION        0x10000000   Versionen löschen
```

siehe auch: ReadUser
WriteUser
UserParent
UserKeys
UserGroups

Property UserGroup (int)

Das Property UserGroup kann kontrolliert werden, ob ein Anwendereintrag eine Gruppe oder einen Anwender beinhaltet..

Parameter:

0	Anwender
1	Gruppe

siehe auch: ReadUser
WriteUser
UserParent
UserKeys
UserFlags

Property UserGroups (AnsiString)

Das Property UserGroups enthält die Gruppen des Anwenders (als Zahlenfolge mit Komma getrennt). Ein Administrator kann dabei beliebige Gruppen zuteilen, ein Subadministrator maximal seine eigenen Gruppen. Es liegt in der Verantwortung des Skript-Programmierers dafür zu sorgen, dass hier wirklich nur Gruppen und keine Anwender oder ungenutzen Ids zugewiesen werden.

siehe auch: ReadUser
WriteUser
UserParent
UserKeys
UserFlags

Property UserId (int)

Über das Property UserId können Sie die Nummer des aktuell aktiven Anwederdatensatzes ermitteln. Obwohl dieses Property auch einen Schreibzugriff erlaubt, sollten Sie diese Nummer nur dann verändern, wenn Sie genau wissen, was Sie durch diesen Aufruf bewirken (im Allgemeinen gibt es für den Scriptprogrammierer keinen Grund diesen Eintrag zu verändern).

Siehe auch: UserName

Property UserKeys (AnsiString)

Das Property UserKeys enthält die Schlüssel des Anwenders (als Zahlenfolge mit Komma getrennt, paarweise die Schlüsselnummer und das Recht). Ein Administrator kann dabei beliebige Schlüssel zuteilen, ein Subadministrator maximal seine eigenen Schlüssel. Es liegt in der Verantwortung des Skript-Programmierers dafür zu sorgen, dass hier wirklich nur verfügbare Schlüssel zugewiesen werden.

Parameter:

Bit 1	L (Lesen)
Bit 2	S (Schreiben)
Bit 3	D (Löschen)
Bit 4	E (Editieren)

Beispiel: 0,5,3,1
Schlüssel Nr.1 mit L+D Rechten
Schlüssel Nr.3 mit L Recht

siehe auch: ReadUser
WriteUser
UserParent
UserKeys
UserFlags

Property UserName (AnsiString)

Über das Property UserName können Sie den Anwendername des aktuell geladenen Anwenderdatensatzes lesen. Dieser Datensatz kann z.B. über ein Event beim Lesen oder Schreiben gefüllt worden sein. In diesem Fall wird das Property ActionKey entsprechend des Aufrufgrundes gesetzt (20000: unmittelbar vor dem Zurückschreiben eines Anwenders, 20001: nach dem Speichern eines Anwenders, 20002: nach dem Lesen eines Anwenders.

Siehe auch: UserId

Property UserParent (int)

Über das Property UserParent, kann einem Anwender ein (Sub)Administrator zugewiesen werden. Dieser (Sub)Administrator hat dann das Recht, die UserDaten des Anwenders zu ändern.

siehe auch: ReadUser
WriteUser
UserGroups
UserKeys
UserFlags

Funktion Version

Gibt die ELO-Versionsnummer in der Form MSSBBB zurück. (Beispiel 123456)

M	Hauptversionsnummer (im Beispiel 1)
SS	Subversion (im Beispiel 23)
BBB	Interne Build Nummer (im Beispiel 456)

Über diese Abfage können Sie sicherstellen, daß die vorhandene ELO Version den benötigten Stand aufweist, falls Sie Funktionen verwenden die erst bei späteren Versionen hinzugefügt worden sind.

int Version()

Parameter:

keine

Rückgabewerte:

Elo-Versionsnummer

Property ViewFileName (String)

Das Property ViewFileName enthält den Namen des im Dokumentenviewer angezeigten Dokuments. Es wird normalerweise in Skripten ausgewertet, die mit dem Ereignis Ereignis "Vor der Anzeige eines Dokuments" verknüpft sind. Innerhalb eines solchen Skripts kann durch Ändern dieses Properties die Anzeige einer anderen Dokumentendatei veranlasst werden.

siehe auch:

Property WindowState (int)

Über dieses Property können Sie den Zustand des ELO Fensters abfragen.

Werte: Normal : 1
Minimiert: 2
Maximiert: 3
Unbekannt: -1

Funktion WriteColorInfo

Mit dieser Funktion können Sie eine Farbdefinition aus dem aktuellen Elo-Farbobjekt schreiben. Die Farbeinstellungen können Sie mittels der Properties ColorInfo und ColorName vorher setzen.

int WriteColorInfo(int ColorNo)

Parameter:

ColorNo: Nummer der zu schreibenden Farbdefinition

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Ungültiger Wert für die Farbnummer
1: ok

siehe auch: ReadColorNo
ColorInfo
ColorName

Funktion WriteKey (int KeyNo, AnsiString KeyName)

Diese Funktion erstellt einen neuen Schlüsseleintrag oder benennt ihn um.

int WriteKey (int KeyNo, AnsiString KeyName)

KeyNo : Schlüsselnummer beginnend mit 1

KeyName : Name des Schlüssels

Rückgabewerte:

-1	Ungültige Schlüsselnummer
0	Fehler
1	Ok

Bsp.:

Erstellt einen neuen Systemschlüssel Nummer 10 mit dem Namen Buchhaltung.
Elo.WriteKey (10,“Buchhaltung“)

siehe auch: ReadKey
ChangeKey

Funktion WriteObjMask

Mit dieser Funktion können Sie eine Maskendefinition schreiben. Die Maske muß vorher mit ReadObjMask gelesen worden sein. Wenn Sie eine neue Maske anlegen wollen, so muß diese mit ReadObjMask(9999) initialisiert werden.

int WriteObjMask( )

Parameter:

keine

Rückgabewerte:

-3: Kein Arbeitsbereich aktiv
-2: Ungültiger Wert für die Maskennummer
-1: Fehler beim Lesen der Datenbank
1: ok

Beispiel:

```
' Anlegen einer neuen Maske mit einer Indexzeile
NEWMASK=9999
Set Elo=CreateObject("ELO.office")

if Elo.ReadObjMask( NEWMASK )<0 then
  MsgBox "Fehler beim Vorbereiten der Maske"
else
  Elo.ObjMName="ELO Testmaske"
  Elo.MaskFlags=25 ' Versionskontrolliert, Recherche+Ablage

  ' eine Indexzeile, Eingabelänge 5..10 Zeichen
  call Elo.SetObjAttribName( 0, "Index 1" )
  call Elo.SetObjAttribKey( 0, "IDX1" )
  call Elo.SetObjAttribMin( 0, 5 )
  call Elo.SetObjAttribMax( 0, 10 )

  x=Elo.WriteObjMask()
  if x<0 then
    MsgBox "Fehler Nr. " & x & " beim Anlegen der neuen Maske."
  else
    MsgBox "Neue Maske mit der Nummer " & Elo.ObjMaskNo & " angelegt."
  end if
end if
```

siehe auch: ReadObjMask
GetObjMaskNo
MaskFlags

Funktion WriteUser

Die Funktion WriteUser schreibt den aufbereiteten Anwenderdatensatz zurück in die Systemdatei. Administratoren können beliebige Anwender schreiben, Subadministratoren können nur eigene Anwender oder neue Anwender schreiben.

int WriteUser()

Parameter:

keine

Rückgabewerte:

-5:	Ein Subadministrator hat versucht einen fremden Anwender zu beschreiben
-4:	Fehler beim Lesen vom AccessManager
-3:	Ein normaler Anwender hat versucht zu schreiben
-2:	Fehler beim Schreiben zum AccessManager
-1:	Keine Anwendernummer aktiv

siehe auch: ReadUser
UserGroups
UserParent
UserKeys
UserFlags

Funktion WriteUserProperty

Die Funktion WriteUserProperty schreibt einen Anwenderzusatz in eines der 8 Anwenderdatenfelder. Beachten Sie bitte, dass die ersten 4 Felder von ELO reserviert sind (z.B. für die NT-Namensumsetzung). Die Felder 5..8 stehen zur freien Verfügung.

Diese Funktion bietet eine Option, dass ein Propertywert sofort abgespeichert wird. Das ist besonders dann notwendig, wenn die Daten von einem Nicht-Administrator gespeichert werden sollen, in diesem Fall steht die Funktion WriteUser nicht zur Verfügung. Diesen Zustand erreichen Sie, indem Sie auf die Propertynummer den Wert 1000 aufaddieren. Wenn Sie das Property 5 beschreiben, wird dieses nur im Client zwischengespeichert und erst über ein WriteUser dauerhaft gespeichert. Das gleiche Property mit 1005 führt den Speichervorgang sofort aus. Diese Option steht nur für die Properties 5..8 zur Verfügung damit ein Anwender keine Systemeinstellungen verändern kann.

int WriteUserProperty(int PropertyNo, AnsiString Property)

Parameter:

PropertyNo	Propertynummer 1..8
Property	Zu schreibende Anwenderdaten

Rückgabewerte:

-1:	Ungültige Propertynummer
-3:	direktes Schreiben ist nur für die Properties 5..8 (1005..1008) erlaubt
-4:	Fehler beim Lesen der Anwenderdaten
1:	Ok

Beispiel:

Über zwei einfache Skripte kann man ELO so einstellen, dass beim Beenden in den Anwenderdaten die aktuelle Archivposition abgespeichert wird. Sobald das Archiv dann neu betreten wird, springt ELO an die entsprechende Stelle.

Hierfür müssen zwei Skripte angelegt werden:

```
'ExitArchive.VBS
Set Elo=CreateObject("ELO.office")
if Elo.SelectView(0)=1 then
  Id=Elo.GetEntryId(-1)
  if Id>1 then
    if Elo.Version>300356 then
      call Elo.WriteUserProperty(1005,Id)
    else
      if Elo.ReadUser( Elo.ActiveUserId )>0 then
        call Elo.WriteUserProperty(5, Id)
        call Elo.WriteUser
      end if
    end if
  end if
end if


'EnterArchive.VBS
```

```
Set Elo=CreateObject("ELO.office")
if Elo.ReadUser( Elo.ActiveUserId )>0 then
  StartObj=Elo.ReadUserProperty(5)
  if StartObj<>"" then
    Elo.GotoId(StartObj)
  end if
end if
```

siehe auch: ReadUserProperty
UserGroups
UserParent
UserKeys
UserFlags

Funktion WriteWv

Schreibt einen Wiedervorlagetermin (bestimmt durch den Parameter WvIdent) aus dem internen Wv-Speicher in die Datenbank. Dieser Termin kann dann mit den verschiedenen Property-Funktionen vorbelegt werden. Ein WvIdent 0 bewirkt, daß ein neuer Wv-Termin angelegt wird. Vor dem Füllen eines neuen Termins muß der interne Wv-Speicher durch ein ReadWv Aufruf mit Parameter 0 gelöscht werden.

int WriteWv(int WvId)

Parameter:

WvId: Nummer des zu schreibenden Termins, 0: neuen Termin anlegen

Rückgabewerte:

-1: Kein Arbeitsbereich aktiv
-2: Fehler beim Schreiben
-3: WvParent nicht gültig
-4: Keine Kurzbezeichnung eingegeben
1: ok

Siehe auch: ReadWv
DeleteWv
WvIdent
WvParent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Property WvCreateDate (AnsiString)

Über das Property WvDate können Sie das Erzeugungs-Datum Wv-Termins lesen oder schreiben. Hier sollte das aktuelle Tagesdatum stehen. Wenn Sie diesen Eintrag frei lassen, wird es automatisch beim Speichern eingesetzt.

Siehe auch:
ReadWv
WriteWv
WvIdent
WvParent
WvUserFrom
WvUserOwner
WvDate
WvPrio
WvParentType
WvShort
WvDesc

Property WvDate (AnsiString)

Über das Property WvDate können Sie das Datum Wv-Termins lesen oder schreiben.

Siehe auch: ReadWv
WriteWv
WvIdent
WvParent
WvUserFrom
WvUserOwner
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Property WvDesc (AnsiString)

Über das Property WvDesc können Sie einen Memotext zu einem Wv-Termin lesen oder schreiben

Siehe auch: ReadWv
WriteWv
WvIdent
WvParent
WvUserFrom
WvUserOwner
WvCreateDate
WvPrio
WvParentType
WvDate
WvShort

Property WvDueDate (AnsiString)

Dieses Property übergibt oder setzt das Datum der Wiedervorlage, an dem das Dokument gesehen wurde. Die Datumsangabe muss sich an dem aktuell im System eingestelltem Format für Datumseingaben orientieren, im Allgemeinen also in der Form TT.MM.JJJJ vorliegen.

Maximale Länge: 12 Zeichen

siehe auch: WvNew
EditWv
WvListInvalidate
WvActionCode

Property WvIdent (int)

Über das Property WvIdent können Sie die interne Elo-Nummer des aktuellen Wv-Termins lesen oder schreiben. Im Normalfall gibt es keinen Grund dazu diesen Eintrag zu verändern, er wird durch ReadWv bzw WriteWv gesetzt.

Siehe auch:
- ReadWv
- WriteWv
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

Funktion WvListInvalidate ()

Diese Funktion aktualisiert die Wiedervorlageliste.

int WvListInvalidate ()

Rückgabewerte:

1	Aktualisierung erfolgreich
-1	kein Arbeitsbereich aktiv

siehe auch: WvNew
EditWv
WvDueDate
WvActionCode

Property WvNew (int)

Dieses Property gibt an ob eine neue Wiedervorlage erstellt wurde.

Werte:

0 – Nein
1 – Ja

siehe auch: EditWV
WvDueDate
WvListInvalidate
WvActionCode

Property WvParent (int)

Über das Property WvParent können Sie die interne Elo-Nummer des Archiveintrags zu dem aktuellen Wv-Termin lesen oder schreiben. Der Archiveintrag kann ein Schrank, Ordner, Register oder Dokument sein.

Siehe auch:
ReadWv
WriteWv
WvIdent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Property WvParentType (int), WvParentTypeEx (int)

Über das Property WvParentType bzw. WvParentTypeEx können Sie die Art des Elo-Eintrags bestimmen welcher mit dem aktuellen Wv-Termins verbunden ist. Bei allen anderen Werten wird dieser Eintrag aus der Datenbank über den Eintrag WvParent ermittelt.
Bei Archiven mit einer vierstufigen Hierarchie wird mit WvParentType gearbeitet, bei Archiven mit mehr als 4 Hierarchiestufen mit WvParentTypeEx.

WvParentType:
1=Schrank, 2=Ordner, 3=Register, 4=Dokument,

WvParentTypeEx:
1=Schrank, 2=Ordner, 3=..., ..., 253=Register, 254=Dokument,

Siehe auch: ReadWv
WriteWv
WvIdent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvParent
WvPrio
WvShort
WvDesc

Property WvPrio (int)

Über das Property WvPrio können Sie die Priorität des aktuellen Wv-Termins lesen oder schreiben. Es stehen die Werte 1 (Wichtig), 2 (Normal) und 3 (weniger Wichtig) zur Verfügung. Alle anderen Werte werden automatisch auf 2 (Normale Priorität) gesetzt.

Siehe auch:
ReadWv
WriteWv
WvIdent
WvUserOwner
WvUserFrom
WvDate
WvCreateDate
WvParent
WvParentType
WvShort
WvDesc

Property WvShort (AnsiString)

Über das Property WvShort können Sie die Kurzbezeichnung eines Wv-Termins lesen oder schreiben. Diese Eingabe ist für einen neuen Termin zwingend notwendig, wenn Sie fehlt wird die Speicheropertion mit einem Fehler abgebrochen.

Siehe auch:
- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserFrom
- WvUserOwner
- WvCreateDate
- WvPrio
- WvParentType
- WvDate
- WvDesc

Property WvUserFrom (int)

Über das Property WvUserFrom können Sie den Absender eines Wv-Termins lesen oder schreiben. Im Normalfalle sollte hier Ihre eigene UserId stehen. Wenn Sie diesen Eintrag leer lassen, wird Ihre Id automatisch eingesetzt.

Siehe auch:
- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserOwner
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

Property WvUserOwner (int)

Über das Property WvUserOwner können Sie den Eigentümer eines Wv-Termins lesen oder schreiben.

Siehe auch: ReadWv
WriteWv
WvIdent
WvParent
WvUserFrom
WvDate
WvCreateDate
WvPrio
WvParentType
WvShort
WvDesc

Anhang

Anhang A, Bezeichnung der Dialogelemente in der ELO Hauptansicht

Wenn Sie mit der Funktion „ClickOn“ arbeiten möchten oder auch ELO Standardfunktionen mit Skripten überlagern möchten, benötigen Sie die Namen der entsprechenden Menüpunkte oder Toolbar-Buttons. In der unten stehenden Auflistung wird zwischen Buttons und Menüpunkten mit Hilfe der Kennungen „B:“ und „M:“ unterschieden.
Die Kürzel innerhalb der Bezeichner haben folgende Bedeutung:
„Arc...“: Archivansicht, „Clip...“: Klemmbrett, „Post...“: Postbox, „Search...“: Suchansicht, „Wv...“: Wiedervorlageansicht, „mm..“: Menüpunkt im Hauptmenü, „Kom...“: Funktion von ELOkommunal, „...Popup...“: Menüpunkt innerhalb eines Kontext-Menüs, „Plp...“: Menüpunkt innerhalb des Kontext-Menüs der Postbox, „...UserBtn...“: Skript-Buttons

Wichtiger Hinweis: unter ELOoffice sind nicht alle Elemente dieser Liste verfügbar.

Menütext oder Hint	Name des Controls
B:Zoom 25%	ArcZoom25
B:Zoom 50%	ArcZoom50
B:Zoom 100%	ArcZoom100
B:Zoom Dokument einpassen	ArcZoomFit
B:Zurück zum Archivanfang	BtArcStart
B:Einen Schritt zurück	BtArcBack
B:Bestehenden Eintrag bearbeiten	BtArcEdit
B:Ausgewählten Eintrag löschen	BtArcErase
B:Ebenenwechsel unterdrücken	BtArcLock
B:Zoom Breite einpassen	ArcZoomWidth
B:Aktuelles Dokument drucken	BtArcPrint
B:90° nach links drehen	ArcRot270
B:Auf den Kopf stellen	ArcRot180
B:90° nach rechts drehen	ArcRot90
B:Seiten zusammengefaßt scannen	ArcScan
B:Quickinfo anzeigen	ArcShowHint
B:Anwenderdefinierte Vergrößerung	ArcZoomUsr
B:Vollbildansicht	BtArcMaximize
B:E-Mail	BtEMail
B:Fax	BtArFax
B:Zoom Cursor	ArcZoom
B:OCR Bereich	ArcZoomOcr
B:Schrank erzeugen	BtNew1
B:Ordner erzeugen	BtNew2
B:Seite an aktives Archivdokument anfügen	ArcScanAdd
B:Adhoc Workflow starten	BtAdhoc
B:Dokument auschecken und bearbeiten	btCheckOut
B:Dokument erzeugen	BtNewDoc
B:Dokument einchecken	btCheckIn
B:Aktuelles Dokument durchsuchen	BtArcSearch
B:Dokument einpassen	ClipZoomFit
B:Zoom 100%	ClipZoom100
B:Zoom 50%	ClipZoom50
B:Zoom 25%	ClipZoom25
B:Aus dem Klemmbrett entfernen	ClipErase
B:Breite einpassen	ClipZoomWidth
B:Aktuelles Dokument drucken	BtClipPrint
B:Anwenderdefinierte Vergrößerung	ClipZoomUsr
B:90° nach links drehen	ClipRot270

B:Auf den Kopf stellen	ClipRot180
B:90° nach rechts drehen	ClipRot90
B:Vollbildansicht	BtClipMaximize
B:Fax	BtKbFax
B:Zoom Cursor	ClipZoom
B:OCR Bereich	ClipZoomOcr
B:E-Mail	BtClipMail
B:Aktuelles Dokument durchsuchen	BtClipSearch
B:Zoom 25%	PostZoom25
B:Zoom 50%	PostZoom50
B:Zoom 100%	PostZoom100
B:Dokument einpassen	PostZoomAll
B:Breite einpassen	PostZoomWidth
B:Seiten versenden	PostSend
B:Postkorbeinträge sammeln	ReloadPL
B:Ausgewählte Einträge löschen	PostErase
B:Seiten scannen	PostScan
B:Seiten zusammenfassen	PostCollapse
B:Seiten trennen	PostExpand
B:Scanner auswählen	PostSelScan
B:90° nach links drehen	PostRot270
B:Auf den Kopf stellen	PostRot180
B:90° nach rechts drehen	PostRot90
B:Aktuelles Dokument drucken	BtPostPrint
B:Seiten zusammengefaßt scannen	PostDirectAppend
B:Anwenderdefinierte Vergrößerung	PostZoomUsr
B:Vollbildansicht	BtPostMaximize
B:Fax	BtPbFax
B:Zoom Cursor	PostZoom
B:OCR Bereich	PostZoomOcr
B:E-Mail	BtPostMail
B:Archivablage über Auswahldialog	btTreePost
B:Zoom 25%	SearchZoom25
B:Zoom 50%	SearchZoom50
B:Zoom 100%	SearchZoom100
B:Dokument einpassen	SearchZoomFit
B:Einträge suchen	BtStartSearch
B:Breite einpassen	SearchZoomWidth
B:Aktuelles Dokument drucken	BtSrcPrint
B:Haftnotizen suchen	BtSearchNote
B:Anwenderdefinierte Vergrößerung	SearchZoomUsr
B:90° nach links drehen	SrcRot270
B:Auf den Kopf stellen	SrcRot180
B:90° nach rechts drehen	SrcRot90
B:Volltextrecherche	BtSearchVt
B:Vollbildansicht	BtSearchMaximize
B:Fax	BtSrFax
B:Zoom Cursor	SearchZoom
B:OCR Bereich	SearchZoomOcr
B:E-Mail	BtSrcMail
B:Dokument auschecken und bearbeiten	btSrCheckOut
B:Baumansicht anzeigen	SrcTree
B:Aus der Trefferliste entfernen	SrcDelete
B:Dokument einchecken	btSrcCheckIn
B:Aktuelles Dokument durchsuchen	BtSearchCOLD
B:Zoom 25%	WvZoom25
B:Zoom 50%	WvZoom50
B:Zoom 100%	WvZoom100
B:Dokument einpassen	WvZoomFit
B:alle Prioritäten anzeigen	WvPrioC
B:Nur höchste Priorität anzeigen	WvPrioA

B:hohe und mittlere Prioritäten anzeigen	WvPrioB
B:Wiedervorlagetermine sammeln	WvCollect
B:Termin löschen	WvErase
B:Breite einpassen	WvZoomWidth
B:Aktuelles Dokument drucken	BtWvPrint
B:Anwenderdefinierte Vergrößerung	WvZoomUsr
B:90° nach links drehen	WvRot270
B:Auf den Kopf stellen	WvRot180
B:90° nach rechts drehen	WvRot90
B:Vollbildansicht	BtWvMaximize
B:Fax	BtWvFax
B:Zoom Cursor	WvZoom
B:Gruppentermine einblenden	btWithGroup
B:Vertretungstermine einblenden	btWithVert
B:Dokument auschecken	btWvCheckOut
B:Erledigt, Workflow fortsetzen	btGoOn
B:Aktuelles Dokument durchsuchen	BtWvSearch
B:Optionen	BtOptions
B:Nachrichten quittieren	BtDelMsg
B:Nachrichten sammeln	BtMsgReread
B:Nachricht versenden	BtSendMail
B:Einträge neu sammeln	mbCollect
B:Büropläne	mbNewPlan
B:Verteilerplan erstellen	mbGeneratePlan
B:Dokument einchecken	mbCheckIn
B:Dokument bearbeiten	mbEdit
B:Dokumentenänderungen verwerfen	mbVerw
B:Zoom Cursor	CIZoomSel
B:Zoom 25%	CIZoom25
B:Zoom 50%	CIZoom50
B:Zoom 100%	CIZoom100
B:Breite einpassen	CIZoomWidth
B:Dokument einpassen	CIZoomFit
B:90° nach links drehen	CIRot270
B:Auf den Kopf stellen	CIRot180
B:90° nach rechts drehen	CIRot90
B:Anwenderdefinierte Vergrößerung	CIZoomUser
B:Aktuelles Dokument drucken	CIPrint
B:Fax	CIFax
B:Aktuelles Dokument durchsuchen	BtCheckSearch
B:Dokumentensperre erneuern	mbNewCheckOut
M:Archiv	Archiv1
M:CD-ROM Veröffentlichung vorbereiten...	mmPrepCDR
M:Export...	mmExport1
M:Import...	mmImport1
M:Archivübersicht drucken...	mmPrintInfo
M:Reports	Reports1
M:Report anzeigen...	mmShowReport
M:Wiedervorlagen-/Postbox-Report...	WiedervorlagenReport1
M:System - Infocenter...	mmInfocenter
M:Systemdiagnose...	mmSysDiag
M:Accessmanager-Diagnose...	mmEloAM
M:Serverstatus	Serverstatus1
M:Ebene tiefer	Ebenetiefer1
M:Ebene zurück	Ebenezurck1
M:Einträge löschen/wiederherstellen	GelschteEintrge1
M:Anzeigen	mmShowDeleted
M:Wiederherstellen...	mmUndelete
M:Dauerhaft entfernen...	mmDropDel

M:Altdokumente entfernen...	mmDelDocs
M:Verfallsdokumente löschen...	mmDropOld
M:Archivschlüssel setzen...	mmArcKey
M:Drucker auswählen...	SelPrinter
M:Jukebox-Sicherung...	mmJBBackup
M:ELOprofessional MOBIL Aktionen...	ELOprofMobilStart
M:Beenden	Beenden1
M:Bearbeiten	Bearbeiten1
M:Dokument...	mmDokumentBearbeiten
M:Verschlagwortung...	mmEntryEdit
M:Alles markieren	mmMarkAll
M:Auf das Klemmbrett legen	mmClipEntry
M:Berechtigungen setzen...	mmDokKey
M:Löschen	mmDeleteEntry
M:Neu anlegen...	NeuerEintrag1
M:Schriftfarbe setzen...	mmMarkEntry
M:Zur Wiedervorlage	mmWvEntry
M:Einfügen	MNInsert
M:Kopieren	Kopieren1
M:Link einfügen	mmAddLink
M:Links zusammenstellen	mmCollectLink
M:Dokumente versenden	mmSendMap
M:Dokumente empfangen	mmRecieveMap
M:Systemverwaltung	Systemverwaltung1
M:Optionen...	mmOptionen
M:Scanner auswählen...	SelScanner
M:Scan Profil auswählen	SelPagesize
M:nach Vorausschau	PgSizePrescan
M:Anwender...	mmEditUser
M:Ablagemasken...	mmEditMask
M:Dokumentenpfade...	mmEditPath
M:Schriftfarbe...	mmEditMarker
M:Reportoptionen...	mmEditReport
M:Schlüssel...	mmEditKey
M:Aktenplan...	mmAZDefinieren
M:Passwort...	mmEditPwd
M:Skript...	Skriptverwaltung1
M:Stichwortlisten	mmEditBuzz
M:Indexzeilen...	mmSwlIndex
M:Versionsnummern...	mmSwlVer
M:Kommentar...	mmSwlComment
M:Vorlagen...	mmTemplate
M:Vertretungsregelung...	mmVert
M:Verschlüsselungskreise...	mmCryptParams
M:Workflow	mmWorkflow
M:Aktive Workflows anzeigen...	AktuelleAblufe1
M:Fristüberschreitungen...	NotifyAlert1
M:Report	mmWFRep
M:Übersicht	Report2
M:Workflow definieren...	Schemaerstellen1
M:Dokument	Dokument1
M:Einzelseiten scannen	mmScanPage
M:Seiten zusammengefasst scannen	mmMultiScan
M:Seite im Archiv anhängen	mmArcScanAdd
M:Datei einfügen...	mmImportPage
M:Datei speichern unter...	GrafikExport1
M:Barcode-Erkennung	mmBarcode
M:Notizen einfügen ??	Notizeneinfgen1
M:OCR Clipboard	mmOcrClip
M:Suchen	Eintrag1
M:Suchen...	mmEntrySearch

M:Volltextsuche	mmSearchVT
M:Haftnotiz suchen	Haftnotizsuchen1
M:Übergabe Postbox	mmToPost
M:Aktivieren/ Anzeigen	mmActivateDoc
M:Auschecken/ Bearbeiten	mmEditActivateDoc
M:Drucken...	Dokumentdrucken1
M:Einchecken	CheckInMain
M:Versenden	mmSendMail
M:Ansicht	Ansicht1
M:Weitere Ansicht	WeitereAnsicht1
M:Ansicht schließen	mmCloseView
M:Freie Anordnung	mmArrangeFree
M:Nebeneinander anordnen	mmArrangeSide
M:Übereinander anordnen	mmArrangeTop
M:Zoom	Zoom1
M:Zoom 25%	Zoom025
M:Zoom 50%	Zoom050
M:Zoom 100%	Zoom100
M:Maximale Breite	ZoomWidth
M:Einpassen	ZoomUser
M:Anwenderdefiniert	ZoomUsr
M:Drehen	Drehen1
M:90°	mmDrehen090
M:180°	mmDrehen180
M:270°	mmDrehen270
M:Vollbild	VollBild1
M:Ansicht	Ansicht2
M:Archiv	mmGotoArc
M:Klemmbrett	mmGotoClip
M:Postbox	mmGotoPost
M:Suche	mmGotoSearch
M:Aufgaben	mmGotoWv
M:Nachrichten	mmGotoMsg
M:Papieraktenverwaltung	mmGotoAkten
M:In Bearbeitung	mmGotoCI
M:Seite zurück	PagePrev
M:Seite vor	PageNext
M:Erste Seite	PageFirst
M:Letzte Seite	PageLast
M:Aktualisieren	mmRefresh
M:Qualität verbessern	mmScaleToGray
M:Sortierreihenfolge	mmSortList
M:Manuell	SortUser
M:Alphabetisch	SortAlpha
M:Invers Alphabetisch	SortAlphaI
M:Ablagedatum	SortADate
M:Abl.Datum (invers)	SortADateI
M:Dokumentendatum	SortDDate
M:Dok.Datum (invers)	SortDDateI
M:Symbolleiste anpassen...	Toolbarskonfigurieren1
M:Hilfe	Hilfe1
M:Hilfe verwenden...	HelpOnHelp
M:Inhalt...	HelpContent
M:Über das Programm...	HelpAbout
M:Gehe zu	SlpGoto
M:Aus der Liste entfernen	SlpRemove
M:Weitere Referenzen	slpRefs
M:Adhoc-Workflow starten ...	srAdHocFlow
M:Dokumentendaten	SlpDokumentendaten
M:Bearbeiten...	SlpEditDoc
M:Auschecken/ Bearbeiten...	mmSlpCheckOutExec

M:Aktivieren/ Ansehen...	SlpViewDoc
M:Drucken	SlpPrintDoc
M:Dokument auschecken	mmSlpCheckOut
M:Versionsgeschichte...	SlpDocHistory
M:Datei speichern unter...	SlpExport
M:Neue Version aus Datei laden...	SearchPopupNewVer
M:Link	SrcLink
M:Link einfügen	SrcMakeLink
M:Liste sammeln	SrcCollectLink
M:Standard-Workflow starten ...	srStdFlow
M:Verschlagwortung bearbeiten...	SlpEdit
M:Wiedervorlagetermin anlegen...	SlpMakeWv
M:Kopie auf das Klemmbrett	SlpClip
M:Aktenstruktur kopieren	Aktenstrukturkopieren3
M:Aktenstruktur einfügen	Aktenstruktureinfgen3
M:Archiveinträge anzeigen...	Archiveintrgezhlen3
M:Dokumentenliste drucken	SlpPrintDocList
M:Optionen...	SlpOptions
M:Report...	Report3
M:Mehrspaltige Anzeige	MehrspaltigeAnzeige
M:Hilfe...	Hilfe2
M:Dokumente löschen	PlpEraDoc
M:Verschlagwortungen löschen	PlpEraText
M:Datei	mmFile
M:Datei einfügen...	PlpImportFile
M:Datei speichern unter...	PlpExportFile
M:Allgemeine Dokumentenvorlage	mmGloTemplate
M:Persönliche Dokumentenvorlage	mmPrivTemplate
M:Dokument bearbeiten...	PlpEditOle
M:Neues Dokument aus Vorlage erstellen	PlpNewOle
M:Verschlagwortung bearbeiten...	PlpEditDoc
M:Drehen der Scanseiten	PlpRotate
M:90° Drehen	PlpRot90
M:180° Drehen	PlpRot180
M:270° Drehen	PlpRot270
M:Dokument einfrieren	Dokumenteinfrieren1
M:Klammern nach Trennseiten	CollapseSeparatedDocs
M:Sortieren...	Sortieren1
M:Verschränken	PlpMergePages
M:Ablagemaske voreinstellen... :	PlpDocType
M:Archivablage über Auswahldialog	PlpTreePost
M:Automatische Ablage durch Indexaufbau...	PlpKeyArchive
M:Direktablage ins Archiv...	PlpDirectArchive
M:Registerablage anonym...	PlpMultiStore
M:An Archiv-Dokument anhängen	AnArchivFileanhngen1
M:An Recherchedokument anhängen	AppendSearch
M:Neue Version an Archivdokument binden	plpNewVersion
M:Barcode-Erkennung	PlpBarcode
M:Kopie in andere Postbox...	PlpCopyPost
M:Versenden in andere Postbox...	PlpSendPost
M:Vertretungs-Postboxen einsehen	mmShowVPost
M:Hilfe...	Hilfe3
M:Berechtigungen setzen ...	ArcPopupSetArcKey
M:Dokumentendaten	ArcPopupDocMenu
M:Bearbeiten...	Bearbeiten2
M:Auschecken/ Bearbeiten...	mmCheckOutExec
M:Aktivieren/ Ansehen...	Ansehen1
M:Drucken...	Drucken1

M:Sortieren...	MNSort
M:Dokument auschecken	mmCheckOut
M:Dokument einchecken	mmCheckIn
M:Versionsgeschichte...	History1
M:Datei speichern unter...	ArcPopupExport
M:Neue Version aus Datei laden...	ArcPopupNewVer
M:Link	ArcLink
M:Link einfügen	ArcMakeLink
M:Liste sammeln	ArcCollectLink
M:Schriftfarbe setzen...	ArcPopupSetArcColor
M:Sortierung	ArcPopupSetArcSort
M:Manuell	AlpSortOrder
M:Alphabetisch	AlpSortAlpha
M:invers Alphabetisch	AlpSortInvAlpha
M:Ablagedatum	AlpSortIDate
M:invers Ablagedatum	AlpSortInvIDate
M:Dokumentendatum	AlpSortXDate
M:invers Dok.datum	AlpSortInvXDate
M:Verschlagwortung bearbeiten...	ArcPopupEditText
M:Weitere Referenzen	ArcPopupRefs
M:Auf das Klemmbrett legen	ArcPopupToClip
M:Eigenschaften...	Archiveintrgezhlen1
M:Kopieren der Ablagestruktur/ Dokumente	Aktenstrukturkopieren1
M:Aktenstruktur einfügen	Aktenstruktureinfgen1
M:Adhoc-Workflow starten ...	ArcPopupStartAblauf2
M:Akte anfordern	ArcPopupKomReq
M:Allgemein	ArcPopupAllg
M:Aktuellen Ordner als Standardregister ablegen	NewStdReg
M:Checksumme prüfen	mmChecksum
M:Dokument-Dateien verschieben	mmMoveDocs
M:Konvertieren	Konvertieren
M:Elo -> Tiff	Elo2Tiff
M:Outlook	mmOutlook
M:Registerinhalt mit Outlook synchronisieren	DocSyncOutl
M:Registerinhalt nach Outlook übertragen	DocPopupOutl
M:Registeranbindung Outlook...	ArcPopupOutl
M:Replikationskreisanzeige	ShowRepl
M:Replikationskreise	ArcPopupRepl
M:Report...	Report1
M:Standardregister einfügen...	ArcPopupInsertStdReg
M:Volltext...	mnVolltext1
M:Volltextinhalt abrufen (erstellt Postboxdatei ..	mnFultextInfo
M:Wiedervorlagetermine...	ArcPopupWv
M:Workflow	Vorgangssteuerung1
M:Eigene aktive Workflows...	mmInfoActive
M:Alle aktiven Workflows...	AlleaktivenFlows1
M:Alle abgeschlossenen Workflows...	AbgeschlosseneFlows1
M:Dateianbindung	ArcPopupAttach
M:Neu...	ArcPopupAddAttach
M:Aktivieren	ArcPopupExecAttach
M:Speichern unter...	ArcPopupSaveAttach
M:Versionsgeschichte...	ArcPopupAttHistory
M:Löschen	ArcPopupDelAttach
M:Dokument einfrieren...	MNFreeze
M:Neue Haftnotiz	ArcPopupNote
M:Allgemeine Haftnotiz...	ArcPopupHaftnotiz1

M:Persönliche Haftnotiz...	ArcPopupPersoenlich
M:Stempel...	ArcPopupStamp
M:Standard-Workflow starten ...	ArcPopupStartAblauf
M:Wiedervorlagetermin anlegen...	ArcPopupToWv
M:Löschen...	ArcPopupRemove
M:Hilfe...	Hilfe4
M:Gehe zu	WvGotoEntry
M:Termin bearbeiten...	WvEditEntry
M:Termin löschen	WvDelEntry
M:Erledigt, Workflow fortsetzen...	WvErledigt
M:Workflow annehmen	WvAnnehmen
M:Workflow anzeigen...	WvAblaufAnzeigen
M:Dokumentendaten	Dokumentendaten1
M:Bearbeiten...	WlpEditDoc
M:Auschecken/ Bearbeiten...	mmWlpCheckOutExec
M:Aktivieren/ Ansehen...	WlpViewDoc
M:Drucken	WlpPrintDoc
M:Dokument auschecken	mmWlpCheckOut
M:Versionsgeschichte...	WlpDocHistory
M:Datei speichern unter...	WvPopupSave
M:Link	WvLink
M:Link einfügen	WvMakeLink
M:Liste sammeln	WvCollectLink
M:Verschlagwortung bearbeiten...	WlpEditData
M:Hilfe...	Hilfe5
M:Gehe zu	ClpGoto
M:Aus der Liste löschen	ClpEra
M:Aktenstruktur kopieren	Aktenstrukturkopieren2
M:Aktenstruktur einfügen	Aktenstruktureinfgen2
M:Archiveinträge zählen...	Archiveintrgezhlen2
M:Dokumentendaten	ClpDocData
M:Bearbeiten...	ClpEditDoc
M:Auschecken/ Bearbeiten...	mmClpCheckOutExec
M:Aktivieren/ Ansehen...	ClpViewDoc
M:Drucken...	ClpPrintDoc
M:Dokument auschecken	mmClpCheckOut
M:Versionsgeschichte...	ClpDocHistory
M:Datei speichern unter...	ClpDocExport
M:Neue Version aus Datei laden...	ClipPopupNewVer
M:Report...	ClpDocReport
M:Verschlagwortung bearbeiten...	ClpEdit
M:Hilfe...	ClpHelp
M:Rückgabe ins Archiv	KomPlpReturn
M:Anforderung löschen	KomPlpDelete
M:Termine bearbeiten	KomPlpEditDates
M:Weiterreichen an ...	KomPlpMove
M:Akte annehmen	KomPlpAccept
M:Annahme verweigern	KomPlpRefuse
M:Gehe zu	cpGoto
M:Einchecken	cpCheckIn
M:Bearbeiten	cpEdit
M:Drucken	cpPrint
M:Verwerfen	cbDel