ELOoffice OLE Automation Schnittstelle

[Stand: 17.12.2015 | Programmversion : 9.01.016]

Hinweis: Die Beschreibung der ELOoffice OLE Automation Schnittstelle ist ein Auszug aus der ELOprofessional OLE Automation Dokumentation. Da die beiden Schnittstellen weitgehend identisch sind, können viele Skripte mit geringem Aufwand angepasst werden. In der ELOoffice OLE Automation Schnittstelle fehlen jedoch alle Aufrufe zu Funktionen, die nur unter ELOprofessional oder ELOenterprise verfügbar sind. Es könnte sein, das vereinzelte Funktionen in dieser Dokumentation vorhanden sind, obwohl sie und ihre Funktionalität in der ELOoffice OLE Automation Schnittstelle nicht zur Verfügung stehen. Wenn Sie solche Punkte finden, teilen Sie diese bitte unserem Support-Team mit. Es ist erreichbar unter support@elo.com. Diese Version des Dokuments lehnt sich an die Version der ELOprofessional OLE Automation Schnittstelle vom 24.06.2015 an.

Aufbau und Anwendung der Automation Schnittstelle

Die ELOoffice 2.1 DDE Schnittstelle ist in der professional Version gegen eine modernere und bessere OLE Automation Schnittstelle ersetzt worden. Es ist nun möglich ELO von außen Kommandos zu senden, Eintragungen vorzunehmen oder Informationen aus ELO abzufragen.

Die Skriptnamen können Sie prinzipiell frei vergeben. Diese müssen nur den Vorschriften für einen Dateinamen gerecht werden. Dabei sind alle Skriptnamen, die mit ELO beginnen für ELO Digital Office reserviert. Sie sollten diese nicht verwenden, da sich sonst Namenskonflikte ergeben können.

Die OLE Automation Schnittstelle von ELO ist eng an die ehemalige DDE Schnittstelle angelehnt. Aus diesem Grunde kann man sie nicht als Objektorientiert sondern eher als Kommandoorientiert bezeichnen.

Die Schnittstelle ist für ELOoffice und ELOprofessional weitestgehend identisch, allerdings sind unter ELOoffice nicht alle Funktionen verfügbar. Lediglich beim Erzeugen des Objektes muss beachtet werden, dass ELOoffice eine andere Kennung als ELOprofessional besitzt. Bei ELOoffice muss das Objekt mit der Kennung ELO.office statt mit ELO.professional erzeugt werden.

Wichtige Information:

Bitte beachten Sie den Charakter der OLE-Automation Schnittstelle:

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

Diese beiden Hauptziele führen dazu, dass mitunter im Projekt 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.

Inhalt

1.1 ELO starten, Archiv auswählen und Anmeldedialog

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

- 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.
- 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:

```
// ELOprofessional starten ...
try
  {
    EloServer= CreateOleObject("ELO.professional");
  }
catch (...)
  {
    // ELOprofessional 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:

```
// ELOprofessional starten ...
try
  {
    EloServer= CreateOleObject("ELO.professional");
  }
catch (...)
  {
    // ELOprofessional 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;
```

1.1.1 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.

1.1.2 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);
```

1.1.3 Liste aller Dokumente im Schrank, Ordner, Register

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.

```
// Erstmal 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
      };
  };
```

1.1.4 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);
```

1.1.5 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 aber Version 1.01.048 mit der Funktion LookupMaskName. Die oben beschriebene Funktion reduziert sich auf einen Aufruf:

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

1.1.6 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;
  };
```

1.1.7 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.professional")

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
```

1.1.8 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++;
  };
```

1.1.9 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);
  };
```

1.1.10 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","");
  };
```

1.1.11 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.professional")
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.

1.1.12 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.

1.1.13 Beispiel: Zugriffspfad auf das aktuell selektierte Objekt ausgeben

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

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
```

1.1.14 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.professional")
ver=Elo.Version
if ver<300220 then
  MsgBox "Dieses Skript benötigt mindestens die Version ELOprofessional
3.00.220"
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
```

1.1.15 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.professional")
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()
```

1.1.16 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.professional")
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
```

1.1.17 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 Kurzbezeichnung
End If
Kom = ActiveDocument.BuiltInDocumentProperties("Comments")   ' Übernahme des
Kommentars
DDate$ = ActiveDocument.BuiltInDocumentProperties("Last Save Time")
DDate$ = DateValue(DDate$)  ' Ermittelt gültiges Datum: tt.mm.jj
Dim ELOServer As Object
Set ELOServer = CreateObject("ELO.professional") ' ELO32 OLE-Server
On Error GoTo ANMELDEN                ' ELO ist im AnmeldeDialog
iOleResult = ELOServer.PrepareObject(0, 4, 0)     ' neuer Eintrag, Dokument,
Standard
On Error GoTo 0
ELOServer.ObjMemo = Kom              ' Dokumenten-Kommentar
ELOServer.ObjXDate = DDate$          ' 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, "ELOprofessional 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, "ELOprofessional 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 bei ELOprofessional anmelden.")
GoTo ENDE

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

ENDE:
End Sub
```

1.1.18 Scripting innerhalb der Ablaufsteuerung

Innerhalb der Ablaufsteuerung können jedem Knoten zwei Skripte zugewiesen werden, die jeweils beim Aktivieren bzw. beim Beenden eines Knotens automatisch abgearbeitet werden. Innerhalb eines solchen Skripts kann auf die Daten des aktuellen Knotens zugegriffen werden, hierzu stehen die Properties

NodeAction, NodeActivateScript, NodeAlertWait, NodeAvailable, NodeComment, NodeFlowName, NodeName, NodeTerminateScript, NodeType, NodeUser und NodeYesNoCondition zur Verfügung (Beschreibung der einzelnen Properties s. unten).

Bevor auf diese Properties zugegriffen wird, muß das Property NodeAvailable abgeprüft werden. Bei einem Wert von 1 stehen Informationen über einen Knoten zur Verfügung, ansonsten nicht. Wird das Skript beispielsweise manuell aufgerufen, also nicht im Kontext eines Ablaufes, können keine Informationen zu einem Knoten geliefert werden. In diesem Fall hat das Property NodeAvailable den Wert 0.

Achtung: Beim Weiterleiten einer Workflow-Aktivität wird zuerst der gesamte Workflow zusammen mit der ELO Indexinformation zum Dokument eingelesen. Danach werden die notwendigen Schritte im Workflow und die damit verbundenen Skripte ausgeführt. Wenn Sie nun in einem der Skripte die Indexinformation verändern, sind die neuen Daten innerhalb dieses Schrittes noch nicht sichtbar. Besonders kritisch kann sich das bei Verzweigungsknoten bemerkbar machen. Wenn die Verzweigung aufgrund des Feldinhaltes "BETRAG" stattfindet, und dieses Feld im dem Ende-Script des Knotens vor der Verzweigung verändert wird, dann wird dieser neue Wert nicht in die Entscheidungsfindung einfließen. Falls das doch notwendig ist, muss im Entscheidungsknoten der Gruppenname des entsprechenden Feldes mit einem vorangestellten Ausrufungszeichen geschrieben werden. Hieran erkennt der Workflow, dass er das entsprechende Feld direkt aus der Datenbank und nicht aus den internen Daten auslesen soll.

Beispiel-Skript, das auf die Daten eines Knotens zugreift:

```
CRLF=Chr(13)&Chr(10)
Set Elo=CreateObject("ELO.professional")
If Elo.NodeAvailable=1 Then

  FlowTxt="Name des Ablaufs: "&Elo.NodeFlowName

  Select Case Elo.NodeType
    Case 1  NTxt="Startknoten"
    Case 2  NTxt="Person"
```

```
    Case 3  NTxt="Aufteilungsknoten"
    Case 4  NTxt="Entscheidungsknoten"
    Case 5  NTxt="Sammelknoten"
  End Select

  if Elo.NodeAction=1 Then
    NTxt=NTxt&" aktiviert."
  Else
    NTxt=NTxt&" beendet."
  End If
  NTxt="Aktion: "&NTxt

  NameTxt="Bezeichnung des Knotens: "&Elo.NodeName

  KommentarTxt="Kommentar: "&Elo.NodeComment

  If Elo.NodeType=4 Then
    BedingungTxt="Bedingung: "&Elo.NodeYesNoCondition
  Else
    BedingungTxt=""
  End If

  ActivateTxt="Skript beim Aktivieren: "&Elo.NodeActivateScript
  TerminateTxt="Skript beim Beenden: "&Elo.NodeTerminateScript

  MsgBox FlowTxt&CRLF&NTxt&CRLF&NameTxt&CRLF&KommentarTxt&CRLF_
         &BedingungTxt&CRLF&ActivateTxt&CRLF&TerminateTxt

End If
```

Ist der aktive Knoten ein Verteilerknoten, kann über ein Skript gesteuert werden, an welche Nachfolger das Dokument weitergeleitet werden soll. Ein solches Skript wird im Feld *Ende-Skript* eines (Verteiler-)Knotens eingetragen:

```
Set Elo=CreateObject("ELO.professional")
ObjID=Elo.NodeObjectID              ' Zugriff auf die Daten...
Elo.PrepareObject ObjID,0,0         '..des im Ablauf befindlichen Dokuments
MsgBox Elo.ObjShort
Rv=Elo.OpenChildNodes               'Zugriff auf die Folgeknoten eröffnen
If Rv=1 Then
  Do
   rv=Elo.GetChildNode          'Folgeknoten lesen...
   If (rv>0) Then              '...solange vorhanden
     If Elo.NodeUser=3 Then    'Nur der Anwender Nr.3 erhält das Dokument...
       Elo.NodeAllowActivate=1
     Else                      '...alle anderen nicht
       Elo.NodeAllowActivate=0
     End If
   End If
  Loop Until rv<0
  Elo.SelectCurrentNode     'Zurückschalten zum aktuellen Knoten
End If
```

1.1.19 Scriptereignis "Beim 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.professional" )

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.

Dieses Script Ereignis ist verfügbar ab der Version 3.00.584, bzw. 4.00.182.

Beim Kopieren und Einfügen von Aktenstrukturen wird jetzt ebenfalls dieses Ereignisskript aufgerufen. Das Property „ActionKey" hat in diesem Fall den Wert 4. Wird im Skript das Property „ScriptActionKey" auf den Wert -30 gesetzt erfolgt ein Abbruch des Einfügevorgang.

Dieses Script Ereignis ist verfügbar ab der Version 8.00.056.

1.1.20 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 (Werte 1001 ... 1005 stehen erst ab der Version 5.00.020 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.professional" )
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
```

1.1.21 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.

Ab ELO Client Version 6.00.160:

1. Beim Outlook Drag&Drop kann man im Client einstellen, ob ein Verschlagwortungsdialog angezeigt werden soll. Damit Scripte "Beim Bearbeiten der Verschlagwortung" erkennen können, ob der Dialog aus so einem Drop Vorgang heraus erzeugt wurde, wird während des Droppens ein Cookie "DropMode" gesetzt. Dieses Cookie kann folgenden Inhalt besitzen:
    - "Archive Outlook Attachment" : Ein Outlook Attachment wurde ins Archiv geschoben.
    - "Archive Outlook Mail" : Eine Outlook Mail wurde in Archiv geschoben. Dieser Eintrag ist zudem mit der konfigurierten Email Maske versehen und hat die Indexzeilen Absender und Empfänger gefüllt.
    - "Intray Outlook Attachment" : Ein Outlook Attachment wurde in die Postbox geschoben.
    - "Intray Outlook Mail" : Eine Outlook Mail wurde in die Postbox geschoben.
    - "Archive File" : Eine Datei wurde per Drag&Drop ins ELO Archiv geschoben.
    - "Intray File" : Eine Datei wurde per Drag&Drop in die Postbox geschoben. Bei dieser Aktion wird allerdings kein Verschlagwortungsdialog aufgerufen, so dass das Event "Beim Bearbeiten der Verschlagwortung" auch nicht aktiviert wird.

Voraussetzung: Die Option „Bei Outlook Drag&Drop Transfer Verschlagwortungsdialog anzeigen“ unter „Systemverwaltung – Optionen – Mail/Signatur“ muss aktiviert sein.

2. Ab Version 7.00.066 Zusätzlich zu Punkt 1) wird beim Drag&Drop das Property ‚ObjMainParent‘ gefüllt.

3. Eine Erweiterung des Script Events "Beim Bearbeiten der Verschlagwortung". Ab dieser Version können beim Weiterleiten eines Workflows zusätzliche Indexzeilen angezeigt und bearbeitet werden. Deshalb kann dieser Dialog auch das Event triggern.

- Wenn in einem Personenknoten Indexzeilen definiert sind, dann wird vor der Anzeige des Weiterleiten Dialogs das Script mit ActionKey 28 aufgerufen. Die Verschlagwortungsdaten sind zu diesem Zeitpunkt geladen (Kurzbezeichnung, Indexzeilen, ...). Zusätzlich kann im "TextParam" die Namen der konfigurierten Indexzeilen ausgelesen und bei Bedarf verändert werden.

  Wenn der Anwender mit "Ok" weiterleitet und Indexzeilen definiert sind und der Anwender eine Eingabe in die Indexzeilen vorgenommen hat, dann kommt am Ende vor dem Speichern der Indexdaten das Script mit ActionKey 29. In TextParam sind wieder die Namen der konfigurierten Indexzeilen gespeichert.

Ab Version 7.00.056 Es kann nun die Weiterleitung abgebrochen werden. Dazu setzt man „ScriptActionKey“ auf den Wert „1“. Um dem Anwender eine Nachricht anzuzeigen, warum die Weiterleitung nicht durchgeführt wird, muss folgende Zeile eingefügt werden:

```
Elo.ActivePostFile = „Noch nicht weiterleiten“
```

Ab Version 7.00.066

- Beim Schließen des Verschlagwortungsdialogs wird nun das Skript Event „Beim Bearbeiten der Verschlagwortung“ mit dem ActionKey 25 aufgerufen. Diese Aufruf findet immer statt, auch wenn der Dialog weder mit Abbruch noch mit ‚OK‘ beendet wurde (z.B. durch Alt-F4).
- Bei der Mehrfachverschlagwortung wird nun ebenfalls das Skript Event „Beim Bearbeiten der Verschlagwortung“
- Beim Doppelklick auf das Stichwortlistensymbol in der Verschlagwortung wird nun in der OLE Schnittstelle ein ‚Enter‘ vor der Eingabe des Stichworts und ein ‚Exit‘ Event nach der Eingabe des Stichworts ausgelöst.

Ab Version 7.00.074

- Maskenliste im Verschlagwortungsdialog reduzieren
  Um dem Anwender eine reduzierte Liste von Verschlagwortungsmasken anzubieten kann man 2 Funktionen verwenden. Eine Funktione zur Abfrage der Maskenliste und Eine zum Löschen von Masken im Verschlagwortungsdialog. Diese Funktionen werden sinnvollerweise im ActionKey 24 (Buttons zur Anzeige festlegen) verwendet. Ein Beispielskript, welches alle Masken entfernt die irgendwo „suche“ im Namen enthalten sieht dann so aus:

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

if Elo.ActionKey = 24 then
  Elo.Textparam="100001"

  for i = 0 to 1000
```

```
    name = Elo.GetMaskName(i)
    if name = "" then
      exit for
    end if

    if Instr(LCase(name), "suche") > 0 then
      Elo.DeleteMaskLine(i)
      i = i - 1
    end if
  next
End If
```

1.1.22 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.

1.1.23 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.professional")

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

1.1.24 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.

Dieses Ereignis ist verfügbar ab der Version 4.00.042

1.1.24.1 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.

1.1.24.2 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.

1.1.24.3 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.

1.1.24.4 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.professional")

' 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
```

1.2 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...).

1.2.1 Beispiel: Anzeige der Kurzbezeichnung und der ersten 11 Indexzeilen

```
<html><head>
<title>ELOprofessional 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>
```

1.3 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.

1.3.1 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.professional")
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
```

1.3.2 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.professional")

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

1.3.3 Stichwortlisten (ELO_BUZZLIST)

Für den Fall, dass hinter einer Indexzeile je nach Anwender oder Umgebung unterschiedliche Stichwortlisten eingeblendet werden sollen, gibt es das Script ELO_BUZZLIST. Über den Parameter ViewFileName teilt der Client dem Script mit, welche Stichwortliste Standardgemäß angezeigt werden würde. Dieser Parameter kann dann vom Script abgeändert werden und die Ansicht somit auf eine andere Stichwortliste umgeleitet werden.

Ab der Version 5.0 liegen die Stichwortlisten in der Datenbank. In diesem Fall wird das Script mit etwas anderen Parametern aufgerufen. Die Kennung für die Datenbankabfrage wird über das Property ScriptActionKey weitergegeben, es hat den Wert 1000 (in der alten Form wurde hier die Indexzeilennummer übergeben). ViewFileName enthält den Gruppenname der Indexzeile und ActionKey zeigt mit 1 oder 2 an, ob der Aufruf aus dem normalen Verschlagwortungsdialog kommt oder aus dem Maskendialog oder Hauptmenü. Durch eine Veränderung des Gruppennamens kann man Scriptgesteuert auf eine andere Stichwortliste umschalten, da das Sammeln der Liste durch diesen Eintrag gesteuert wird.

Beachten Sie bitte, dass es in der Version 5.0 keine eindeutige Unterscheidung mehr zwischen Auswahl und Bearbeiten der Liste gibt. Wenn der Anwender das Recht zum Bearbeiten der Stichwortliste besitzt, dann kann er neue Stichworteinträge nun auch immer direkt in der Auswahl hinzufügen.

Beispiel 4.0:

```
Option Explicit
Dim Elo

'-----------------------------------------------
Sub Main
  Dim iRet
  Dim sGroup
  Dim aGNum
  Dim iNum, i
  Dim sFileName

  Set Elo=CreateObject("ELO.professional")

  If Elo.ViewFileName="ELOVER.SWL" then
    iRet=Elo.ReadUser( Elo.ActiveUserId )
```

```
    sGroup=Elo.UserGroups
    If sGroup="" Then Exit Sub
    aGNum=Split(sGroup,",",-1,1)
    iNum=UBound(aGNum)
    If iNum=0 Then
      Elo.ViewFileName=aGNum(0) & "_" & Elo.ViewFileName
      Exit Sub
    End If
    ReDim aGName(iNum+1)
    iRet=Elo.CreateAutoDlg("Gruppen-Stichwortliste wählen:")
    For i=0 To iNum
      iRet=Elo.AddAutoDlgControl(3, 1,Elo.LoadUserName(aGNum(i)),"0")
    Next

    iRet=Elo.ShowAutoDlg
    If iRet=0 Then Exit Sub
    sFileName=""

    For i=0 To iNum
      If Elo.GetAutoDlgValue(i+1)="1" Then
        sFileName=aGNum(i) & "_" & Elo.ViewFileName
      End If
    Next

    If sFileName<>"" Then
      Elo.ViewFileName=sFileName
    End If

  End If
End Sub

'------------------------------------------------
'------------------------------------------------
Main

Beispiel 5.0

Set Elo=CreateObject("ELO.Professional")

if Elo.ScriptActionKey = 1000 then
  UserName = Elo.LoadUserName( Elo.ActiveUserId )
  MsgBox Elo.ViewFileName & vbcrlf & UserName
  Elo.ViewFileName = Elo.ViewFileName & "." & UserName
End if
```

1.3.4 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.

Ab ELO Version 8 ist noch der ‚ActionKey = 3' hinzugekommen. Dieser wird etwas später als der ActionKey 1 gefeuert nachdem der Client einige interne Operationen abgeschlossen hat. Mit Hilfe dieses AktionKey kann man z.B. die Ansicht wechseln oder die Skriptbuttons in der Multifunktionsleiste einbauen.

1.3.5 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.

1.3.6 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.

1.3.7 Suchansicht einstellen (ELO_SEARCHTREE)

Wenn in der Suchansicht die Option "Baumansicht anzeigen" gewählt ist, wird nach der Anzeige des Dialoges "Virtuellen Baum auswählen" das Skript aufgerufen.

ActionKey = 1 - wenn die Standardordnerstruktur gewählt wurde (Ordnerstruktur)

ActionKey = 2 - wenn eine selbst definierte Struktur gewählt wurde

In diesem Beispiel wird die Baumstruktur in einer Messagebox ausgegeben:

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

If Elo.ActionKey = 1 Then
  ' Standardstruktur wird angezeigt
ElseIf Elo.ActionKey = 2 Then

  iRes = Elo.SortParamlist(1)
  For i=1 To Elo.GetParamCount
    s=s & Elo.GetFromParamList(i) & vbCRLF
  Next
  MsgBox "Liste der Treeknoten" & vbCrLf & s

Else
  MsgBox "ActionKey <> 1 and <> 2  -> ungültig"
End If
```

GetParamCount Der Wert enthält die Anzahl der zurückgelieferten Zeilen

GetFromParamList Eine Zeile aus der 'Paramlist'

Parameter enthält die Zeilennummer

ObjType des Dokumentes

ObjShort des Dokumentes

ObjShort der Strukturelemente

1.3.8 Statisches Script Event beim Speichern der Verschlagwortung (ELO_SAVEINDEX)

In vielen Fällen muss beim Speichern der Verschlagwortung eine einfache Aktion durchgeführt werden, z.B. soll die Kurzbezeichnung aus den Indexzeilen gefüllt werden. Das kann man per Script im Event „Beim Bearbeiten der Verschlagwortung“ durchführen. Dieses Scriptevent ist allerdings recht aufwendig, da es auf viele unterschiedliche Ereignisse reagiert und es muss bei den Anwendern konfiguriert werden.

Für solche Fälle gibt es in der Version 7.0 ein statisches Scriptevent „ELO_SAVEINDEX". Wenn es also eine Skriptdatei mit dem Namen „ELO_SAVEINDEX.VBS" gibt, dann wird das Skript ausgeführt beim Speichern der Verschlagwortung (OK Button), ohne dass es irgendwo in der Anwenderkonfiguration angemeldet werden muss. Dieses Scriptevent wird nach dem Event „Beim Bearbeiten der Ver-schlagwortung" mit ActionKey „Beim Speichern" aufgerufen.

Als Beispiel – wenn der Eintrag die Maskennummer 20 hat und die Kurzbezeichnung noch leer ist, dann soll sie mit einer Kombination aus festen Texten und Teilinhalten der Indexzeilen gefüllt werden:

```
Set Elo=CreateObject("Elo.Professional")
if Elo.ObjMaskNo = 20 and Elo.ObjShort = "" and Elo.ActionKey=21 then
Elo.ObjShort = "Dok.: " & Elo.GetObjAttrib(1) & " : " &
Left(Elo.GetObjAttrib(0), 5)
end if
```

Hinweis:

Das Skript wird direkt nach dem Aufruf des Skripts "Beim Bearbeiten der Verschlagwortung – Beenden" aufgerufen.

1.3.9 Eigene Reports

Im Kontextmenü der Archivansicht können Sie aus verschiedenen voreingestellten Aktivitätenreports auswählen. Zusätzlich können Sie hier noch bis zu 4 eigene Reports erstellen. Hier sind alle Reports denkbar, die Sie über eine WHERE-Klausel per SQL Befehl ansteuern können.

Die eigenen Reports werden über ein Script ausgewählt. Hierzu müssen Sie ein Eintrag unter dem Namen ELO_ACTSELECT anlegen. Dieses enthält für jeden der 4 möglichen Reports jeweils zwei Aktionen: a) den Namen im Menü ausgeben und b) den SQL Befehl zusammensetzen. Hierzu werden im Propery "ActionKey" die Werte 1..4 für a) und die Werte 101..104 für b) übergeben. Das Script liefert im Property ViewFileName den gewünschten Wert zurück.

Beispiel:

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

select case Elo.ActionKey
  case 101
    Elo.ViewFileName="Empf m.thiele"

  case 1
    Elo.ViewFileName="destination like 'm.thiele'"

  case 102
    Elo.ViewFileName="Heute zurück"

  case 2
    ToDay=Date
    IsoToDay=Right(ToDay,4)+Mid(ToDay,4,2)+Left(ToDay,2)
    Elo.ViewFileName="backat like '"&IsoToDay&"'"
```

```
end select
```

1.3.10 Skripte aufrufen

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

1.3.10.1 -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.

1.3.10.2 -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.

1.3.10.3 -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.

1.3.10.4 -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

1.4 Formularerkennungs-API

1.4.1 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 ab der Version 2.05.104.

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 (auch neu ab der Version 2.05.104) 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.

1.4.2 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.

1.4.3 Beispiele

1.4.3.1 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:

Ein beliebiger, möglicherweise auch leerer, Vorspann aus beliebigen Zeichen (z.B. weil in das Rechteck durch unsauberen Scannereinzug von oben fremde Zeichen hereinragen).

Der Text ‘Rechnung’

Eine beliebig lange, möglicherweise auch leere, Folge von Leerzeichen

Eine Nummer (die Rechnungsnummer)

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.

1.4.3.2 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
```

1.4.3.3 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.professional")
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
```

1.4.4 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.

1.4.5 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.

1.5 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.

1.6 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.

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
	Beim Schließen der Maske (letzes gefeuertes Event)	25
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
Beim Bearbeiten der Verschlagwortung – Anzeige von Indexzeilen im „Weiterleiten Dialog“ eines Workflows	WF: Vor der Anzeige des Weiterleiten Dialogs	28
	WF: Nach der Anzeige des Weiterleiten Dialogs	29

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.professional")

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
```

1.7 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

1.8 Property ActiveUserId (int, nur lesen)

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

Siehe auch:

- LoadUserName
- FindUser
- SelectUser

1.9 Property Activity (AnsiString)

Das Property Activity setzt oder liest den aktuellen Wert einer Aktivität. Dieser Wert ist nur in der Eventroutine "Beim Lesen oder Schreiben einer Aktivität" gültig. Wenn der Wert zu anderen Zeiten abgefragt wird, kann es zu Fehlern bis hin zu einer Schutzverletzung kommen.

Der Wert der Aktivität ist ein String, welcher alle Felder enthält. Diese sind jeweils durch das Trennsymbol (i.A. ¶) abgegrenzt.

Verfügbar seit: 3.00.510

Beispiel:

```
' Beim Schreiben einer Aktivität wird das Kommentarfeld
' mit "Test99" gefüllt,
' beim Lesen wird die Textdarstellung der Aktivität in
' einer MessageBox angezeigt.
Set Elo=CreateObject("ELO.professional")

if Elo.ActionKey=0 then
  act=split( Elo.Activity, "¶" )
  act(15)="Test99"
  Elo.Activity=join(act,"¶")
end if

if Elo.ActionKey=1 then
  MsgBox Elo.Activity
end if
```

1.10 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.

Beispiel:

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

1.11 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

1.12 Funktion AddNote

1.13 Funktion AddNoteEx

1.14 Funktion AddNoteEx2

1.15 Funktion AddNoteEx3

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 )
int AddNoteEx2( int ObjId, int NoteType, AnsiString NoteText, int PageNo, int x,
int y, int color )
int AddNoteEx3( int ObjId, int NoteType, AnsiString NoteText, int PageNo, int x,
int y, int color, int w, int h, AnsiString Acl )
```

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
Color:            Farbe der Haftnotiz
w:    Breite der Notiz
h:    Höhe der Notiz
Acl:        reserviert für künfige Erweiterung
```

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

Beispiele:

```
' --- Anlegen einer "HaftNotiz"
iRes = Elo.AddNote(119, 1, "Gelbe Haftnotiz nur auf dem Balken")
' --- Anlegen einer "farbigen Haftnotiz"
iRes = Elo.AddNoteEx2 (119, 1, "Grüne Haftnotiz", 0, 0, 0, 65280)
' --- Anlegen einer "Annotation - Haftnotiz mit Position"
iRes = Elo.AddNoteEx (119, 1, "Gelbe Annotation per Skript 1", 1, 200, 600)
' Anlegen einer "Notiz - Nur Annotation mit Position + Größe + Farbe Rot"
```

```
iRes = Elo.AddNoteEx3 (119, 20, "8-1F0000080Arial     Notiz per Skript 1", 1,
200, 800, 255, 400, 200, "")
' Anlegen eines gefüllten Rechtecks
iRes = Elo.AddNoteEx3 (119, 14, "", 1, 200, 800, 255, 400, 200, "")
' Anlegen eines Textmarkers
iRes = Elo.AddNoteEx3 (119, 10, "", 1, 200, 800, 255, 400, 200, "")
' Anlegen leeren Rechtecks
iRes = Elo.AddNoteEx3 (119, 15, "", 1, 200, 800, 255, 400, 200, "")
' Anlegen eines Stempels mit dem Text "Ein Stempel mit Text"
iRes = Elo.AddNoteEx3 (119, 19, "8-1F0000080Arial     Ein Stempel mit Text", 1,
200, 800, 255, 400, 200, "")
```

Formatierungsanweisungen entnehmen Sie bitte der Tabelle „Haftnotiz“ aus der Spalte „pidesc“. Einfach manuell ein Beispiel erzeugen und schauen, was ELO in die Datenbank geschrieben hat.

Wie werden die Farben gesetzt:

'Farbwert setzt sich wie folgt zusammen:

'Farben blau grün rot

'HEXWert FF FF FF

'Den HexWert dann in Integer umwandeln:

'Farbe rot: FF -> 255

'Farbe grün: FF00 -> 65280

'Farbe blau: FF0000 -> 16711680

'Also nach rechts mit 'Nullen' auffüllen.

Farbe Cyan: FFFF00 -> 16776960

Verfügbar ab:

AddNoteEx ab 4.00.138

AddNoteEx3 ab 6.00.116

1.16 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.professional")

' 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

1.17 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
...
```

Verfügbar ab: 4.00.180

1.18 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.professional")

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", ".", " - " )
...
```

Verfügbar ab: 5.00.066

1.19 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.professional")

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
```

Verfügbar ab: 4.00.214

1.20 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

1.21 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.

1.22 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

Verfügbar seit: 5.00.036

Beispiel:

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

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

1.23 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

Verfügbar seit: 5.00.036

Beispiel:

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

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

1.24 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.

Verfügbar seit: 5.00.042

Beispiel:

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

Siehe auch:

- MaskKey
- DocKey
- DocKind
- DocPath

1.25 Property AutoDlgResult (AnsiString)

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

Beispiel:

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

Verfügbar ab: Ver 3.00.228

Siehe auch:

- CreateAutoDlg
- AddAutoDlgControl
- ShowAutoDlg
- GetAutoDlgValue

1.26 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

1.27 Funktion ChangeObjAcl

Verfügbar ab: 3.00.???

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	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)

Ansonst Neue Objektrechte

Beispiel:

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)
```

1.28 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

1.29 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.professional")
MsgBox Elo.CheckFileHash( "Testdatei", "d:\temp\Scandatei.tif", 128 )
```

Verfügbar seit: 4.00.034

Siehe auch:

- GetMD5Hash
- LookupHistMD5

1.30 Funktion CheckIn

1.31 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 ELOprofessional 3.0 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..-16 CIO_LockError, CIO_PathError, CIO_CopyError, CIO_ExecError,
- CIO_UknError, CIO_NoDocument, CIO_ModeError,
- CIO_UserAbort, CIO_ArchiveError, CIO_ReadOnly,
- CIO_DeleteError, CIO_IsLocked, CIO_NoTemplate,
- CIO_ScriptAbort, CIO_IsDeleted, CIO_NoRevs

Verfügbar (Ex) 3.00.278

Beispiel:

Beim Einchecken kann das Dokument verschlüsselt werden.

Achtung: Die alten Versionen des Dokumentes bleiben weiterhin unverschlüsselt und können ohne ‚Passwort' angesehen werden.

```
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 Left(FName,1)<>"-" 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
```

1.32 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

1.33 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

1.34 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 )
```

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)

Verfügbar seit: 4.00.178

Beispiel:

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

1.35 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

1.36 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%.

Verfügbar: 3.00.220

1.37 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

1.38 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.

Wird die Funktion "ClickOn" mit einer Funktion aus dem Kontextmenü verwendet, muss dem Property PopupObjID die Objekt-ID des zu bearbeitenden ELO-Eintrags zugewiesen 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

1.39 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

1.40 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

1.41 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)

Verfügbar seit: 3.00.378

1.42 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

1.43 Property ColorName (AnsiString)

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

Maximale Länge: 30 Zeichen

Siehe auch:

- ReadColorInfo
- WriteColorInfo
- ColorInfo

1.44 Property ColorNo (int)

Das Property ColorNo bestimmt die Farbnummer einer Farbdefinition.

Siehe auch:

- ReadColorInfo
- WriteColorInfo
- ColorName

1.45 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.

Verfügbar seit: 3.00.228

Beispiel:

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

Siehe auch:

- AddAutoDlgControl
- ShowAutoDlg
- GetAutoDlgValue

1.46 Funktion CreateCounter

Die Funktion CreateCounter erzeugt einen neuen Zähler mit einem voreinstellbaren Startwert.

```
int CreateCounter( AnsiString CounterName,  int InitialValue )
```

Parameter:

- CounterName Name des zu erzeugenden Zählers
- InitialValue Startwert

Rückgabewerte:

- -1: Fehler
- 1: Ok.

Siehe auch:

- GetCounterList
- GetCounter

1.47 Funktion CreateStructure (AnsiString Path, int StartID)

Verfügbar ab: Ver 3.00.228

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.

Ab Version 8.00.056

Ein ‚abschließendes' Pilcrow (¶) wird jetzt ignoriert. Bei manchen BP führte es zu Problemen wenn der ‚Index String' aus Variablen zusammen gesetzt wurde und eine der Variablen keinen Wert enthielt.

```
(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)

- -6 - Recht "Archiv bearbeiten" fehlt (ab Version 6.00.078)
- -7 - Anwender besitzt nicht das Recht "Listen bearbeiten" in einem bereits vorhandenen Struktuelement des angegebenen Pfads (ab Version 6.00.078)
- -5 – Kein Eintrag ausgewählt
- -4 – Keine gültige Startposition angegeben
- -3 – Leerer Feldname
- -2 – Kein Arbeitsbereich aktiv
- -1 – Zu viele Objekte (Archivtiefe wird überschritten)

Beispiel:

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)

1.48 Funktion CreateViewer

Über die Funktion können Sie aus einem Exportdatensatz einen Viewerdatensatz erstellen lassen. Hierzu muss das ViewerPostbox Verzeichnis unter ELOprofessional 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.professional")
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

Verfügbar seit: 3.00.354

1.49 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

Verfügbar seit: 3.00.196

Siehe auch:

- IntToDate

1.50 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

1.51 Funktion DeleteDocumentVersions

Der Befehl löscht alle Versionen außer der aktuellen Arbeitsversion und den Milestoneversionen.

```
int DeleteDocumentVersions(int lObjectId, AnsiString Reserviert)
```

Parameter:

- lObjectId: logische ELO Objekt Id, es muss sich um ein Dokumentenobjekt handeln, Ordner haben ohnehin keine Versionen
- Reserviert: leer (hier kann man später mal eine Erweiterung zur optionalen Eingabe bestimmter Versionen einfügen)

Rückgabewerte:

- -1: kein Arbeitsbereich aktiv
- -2: ObjectId zeigt nicht auf ein Dokument
- -3: Fehler beim Lesen der Datenbank
- -4: Kein Löschrecht für Dokumentenversionen vorhanden
- -5: Kein Lese-, Lösch- oder Editrecht
- >=0: Anzahl der gelöschten Versionen

Verfügbar seit:7.00.058

1.52 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.professional")

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
```

Verfügbar seit: 4.00.210

Siehe auch:

- DeleteObj

1.53 Funktion DeleteNote

Die Funktion löscht eine Haftnotiz, auf die zuvor mit FindFirstNote oder FindNextNote zugegriffen wurde.

```
int DeleteNote()
```

Parameter:

- keine

Rückgabewerte:

- 1: Haftnotiz wurde gelöscht
- -1: kein Arbeitsbereich aktiv
- -2: keine Haftnotiz selektiert
- -3: Fehler beim Löschen
- -4: keine ausreichenden Rechte zum Löschen von Haftnotizen vorhanden
- -5: die zu löschende Haftnotiz ist ein Stempel
- -6: die zu löschende Haftnotiz wird gerade von einem anderen User bearbeitet

Verfügbar seit: 6.00.090

Siehe auch:

- FindFirstNote
- FindNextNote
- UpdateNote

1.54 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

1.55 Funktion DeleteProjectOptions

Löscht alle Optioneneinträge eines Projekts in den Vorgaben für die Aktivitäten. Evtl. vorhandene Projektaktivitäten werden aber nicht gelöscht und können weiterhin angezeigt und aufgelistet werden.

Sie können auch gezielt einen einzelnen Eintrag herauslöschen. In diesem Fall müssen Sie statt des Projektnames einen String aus Name, Major-Nr., Minor-Nr. und Wert, getrennt durch den normalen Delimiter (¶) eingeben. Das Wert Feld kann leer bleiben, die Major und Minor Nummern können auf 0 gesetzt werden, wenn Sie nicht bekannt sind. Dann werden aber gleich ganze Gruppen gelöscht, also vorsichtig mit dieser Option umgehen.

```
int DeleteProjectOptions ( AnsiString ProjectName )
```

Parameter:

- ProjectName: Name des zu löschenden Projekts oder Projekteintrags

Rückgabewerte:

- -1: Kein Arbeitsbereich aktiv
- -2: Fehlender oder ungültiger Projektname
- -3: Fehler beim Löschen
- -5: Projekt war nicht vorhanden
- -6: Projekteintrag nicht korrekt formatiert
- -7: Projekteintrag war nicht vorhanden
- 1: ok

Verfügbar ab: 3.00.360

Beispiel:

```
set Elo=CreateObject("ELO.professional")
res = Elo.DeleteProjectOptions( "TEST" )
if res<0 then
  select case res
    case -5
      MsgBox "Projekt wird neu angelegt"
      res=1
    case -8
      MsgBox "Sie haben keine Berechtigung zum Bearbeiten der Daten"
    case else
      MsgBox "Fehler : " & res
  end select
end if
if res>0 then
  call Elo.InsertProjectOptions( "ELO_SYSTEM", 1, 0, "TEST" )
  call Elo.InsertProjectOptions( "TEST" , 10, 1, "Ansprech" )
  call Elo.InsertProjectOptions( "TEST" , 10, 4, "Meier" )
  call Elo.InsertProjectOptions( "TEST" , 10, 2, "Müller" )
  call Elo.InsertProjectOptions( "TEST" , 10, 3, "Schulze" )
end if
```

Siehe auch:

- InsertProjectOption

1.56 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
- -4: Stichwortliste ist derzeit für einen anderen Anwender gesperrt
- 1: Ok

Verfügbar ab: 5.00.066

Beispiel

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

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", ".", " - " )
...
```

1.57 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

1.58 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

1.59 Property DelOutlookName (AnsiString)

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

1.60 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

1.61 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
- sDlgitle: 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

1.62 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

1.63 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.

Verfügbar seit: 5.00.042

Siehe auch:

- MaskKey
- DocKey
- DocKind
- DocPath

1.64 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

1.65 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

1.66 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

1.67 Property DocTPath (int)

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

Siehe auch:

- DocPath

1.68 Funktion DoEditObject

1.69 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)
int DoEditObject()
```

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.professional" )

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

1.70 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

1.71 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

1.72 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

1.73 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 Recherche liste 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).

```
int DoInvisibleSearch()
```

Parameter:

- Keine

Rückgabewerte:

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

Verfügbar: 3.00.188

Beispiel:

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

'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
```

Siehe auch:

- PrepareObject
- SelectLine
- DoFulltextSearch

1.74 Funktion DoSearch / DoSearchSel(AnsiString) / 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:

- -2: Allgemeiner SQL-Fehler
- -1: Kein Arbeitsbereich aktiv
- ansonsten: Anzahl der gefundenen Einträge.

Verfügbar seit: DoSearchEx: 3.00.358

ObjFlags setzen: Bit 29: 1 Bei einer Suche wird die Versionsgeschichte mit durchsucht

Suche nach ObjektTyp einschränken:

Suche nur nach Dokumenten Elo.ObjTypeEx = 9998

Suche nur nach Ordnern: Elo.ObjTypeEx = 9999

Beispiel:

```
' führt drei Suchen nach den Begriffen ELO, Test und Word
' aus und zeigt das Ergebnis in einer Trefferliste an
```

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

call Elo.PrepareObjectEx(-3,0,0)
Elo.ObjShort="elo"
call Elo.DoSearchEx( "", 1 )

call Elo.PrepareObjectEx(-3,0,0)
Elo.ObjShort="test"
call Elo.DoSearchEx( "", 0 )

call Elo.PrepareObjectEx(-3,0,0)
Elo.ObjShort="Word"
call Elo.DoSearchEx( "", 0 )
```

Siehe auch:

- PrepareObject
- SelectLine
- DoFulltextSearch

1.75 Funktion DoSelArcTree

1.76 Funktion DoSelArcTreeEx

1.77 Funktion DoSelArcTree3

Öffnet einen Dialog zur Auswahl eines ELO-Eintrags.

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

int DoSelArcTreeEx(int hwndParent,
AnsiString sDlgTitle,
int nCtrl,
int nMinimize,
int ObjId);

int DoSelArcTree3(int hwndParent,
AnsiString sDlgTitle,
int nCtrl,
int nMinimize,
int ObjId,
int nRoot);
```

Parameter:

- HwndParent: Windows-Handle des übergeordneten Fensters oder NULL
- sDlgTitle: Dialogtitel
- nCtrl: 1 für nur Register können ausgewählt werden
- nMinimize: <> 0 für ELO minimieren.
- ObjId: DoSelArcTreeEx öffnet den Tree an der übergebenen ObjektID
- nRoot: DoSelarcTree3 zeigt einen TeilTree ab der übergebenen ObjektID (nRoot) an und öffnet diesen TeilTree an der übergebenen ObjektID (ObjID).

Rückgabewerte:

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

1.78 Funktion EditActivity (AnsiString)

Diese Funktion bearbeitet eine Aktivität. Sie können über den Eingabestring eine Vorbelegung vornehmen, insbesondere wenn Sie zu einem Dokument eine neue Aktivität anlegen, müssen Sie die EloGuid mit der GUID des Dokuments initialisieren. Sofern möglich, sollten Sie auch den Projekt-Eintrag vorbelegen.

Der ActInfo-String setzt sich aus folgenden Teilen zusammen:

```
ActGuid           Eindeutige Kennung der Aktivität, bei einem Neueintrag leer
lassen!
DocGuid     GUID des ELO-Dokuments (per GetGuidFromObj ermitteln).
Destination Empfänger
RevVers           Version
ActTStamp   Zeitstempel, leer lassen, wird beim Speichern automatisch erzeugt
Project           Projektname, nach Möglichkeit vorbelegen
Owner       Eigentümer, ELO Anwendernummer
Creator           Erzeuger, ELO Anwendernummer
Prio        0,1 oder 2
ShortDesc   Kurzbezeichnung
SentAt            Versendedatum ISO-Format
SentMode    Versandart
DueDate     Erwartetes Rückgabedatum, ISO-Format
BackAt            Rückgabedatum, ISO-Format
BackMode    Rückgabestatus
Comment     Zusatztext
FileName    Versendedateiname
UD0   Anwenderdefiniertes Feld 1
UD1
...
UD9   Anwenderdefiniertes Feld 10
```

AnsiString EditActivity(AnsiString ActInfo)

Parameter:

- ActInfo: Aktuelle Feldbelegung, die einzelnen Texte jeweils durch das Trennsymbol (¶) verbunden.

Rückgabewert:

- Leer Es ist ein Fehler aufgetreten oder der Anwender hat den Dialog mit "Abbruch" beendet.

Verfügbar seit: 3.00.360

Beispiel:

```
Id=Elo.GetEntryId(-1)
if (Id>1) then
  guid=Elo.GetGuidFromObj(Id)
  res="¶" & guid
```

```
  res=Elo.EditActivity( res )
  MsgBox res
  Elo.WriteActivity( res )
end if
```

1.79 Property EditDlgActive (int)

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

Rückgabewerte:

- 0 – Nein
- 1 – Ja

1.80 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
-

Siehe auch:

- WVNew
- WvDueDate
- WvListInvalidate
- WvActionCode

1.81 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

1.82 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)
// Was passiert bei Export von verschl. Doks ohne gültiges Passwd
#define IEX_EXPORT_CRYPT_QUERYPASSWD   (1 << 13)
#define IEX_EXPORT_CRYPT_SKIP   (1 << 14)
// Referenzen durch Originale ersetzen
#define IEX_REFS_AS_ORGS   (1 << 15)
// Stichwortlisten exportieren
#define IEX_BUZZWORDS   (1 << 16)
```

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.

1.83 Funktion FindFirstNote

Mit Hilfe der Funktionen FindFirstNote und FindNextNote kann auf die Haftnotizen eines Dokuments zugegriffen werden. Vor dem Aufruf von FindFirstNode muss die Verschlagwortungsinformation mit PrepareObjectEx eingelesen werden.

```
int FindFirstNote()
```

Parameter:

- -

Rückgabewerte:

1: Haftnotiz wurde gefunden

0: keine Haftnotiz gefunden

Verfügbar seit: 6.00.090

Beispiel:

Alle Haftnotizen zum Dokument mit der Objekt-ID iID ermitteln und anzeigen:

```
Set Elo=CreateObject("ELO.professional")
Elo.PrepareObjectEx iID,0,0
iCount=0
sTxt=""
iRes=Elo.FindFirstNote
Do While iRes=1
  iCount=iCount+1
  sTxt=sTxt & "Haftnotiz Nr." & iCount & ": " & Elo.NoteText & vbCRLF & vbCRLF
  iRes=Elo.FindNextNote
Loop
MsgBox sTxt
```

Siehe auch:

- FindNextNote
- UpdateNote
- DeleteNote

1.84 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

1.85 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

1.86 Funktion Funktion FindUser / 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

Verfügbar seit: 5.00.162 für FindUserEx

Siehe auch:

- ActiveUserId
- LoadUserName
- SelectUser

1.87 Funktion FreezeDoc / FreezeDocEx

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 )
Int FreezeDocEx( int ObjectId, int iPrinter )
```

Parameter:

- ObjectId: logische Nummer des zu konvertierenden Dokuments
- iPrinter: logische Nummer für den Drucker
- 1 = TIFF Konvertierung
- 2 = PDF Konvertierung

Rückgabewerte:

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

Beispiel:

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

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
```

Verfügbar seit: 3.00.348

1.88 Funktion FromClipboard

Diese Funktion übernimmt einen Text vom Windows Clipboard

```
AnsiString FromClipboard()
```

Parameter:

- keine

Rückgabewerte:

- Text aus dem Windows Clipboard

Verfügbar seit:3.00.456

Siehe auch:

- ToClipboard

1.89 Funktion GetArcKey

Diese Funktion ermittelt den aktuellen Archivschlüssel.

```
int GetArcKey(  )
```

Parameter:

- keine

Rückgabewerte:

- Archivschlüssel oder -1 bei Fehler

1.90 Funktion GetArcName

Diese Funktion ermittelt den aktuellen Archivnamen.

```
AnsiString GetArcName(  )
```

Parameter:

- keine

Rückgabewerte:

- Archivname oder Leerstring bei Fehler

1.91 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

1.92 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.

Beispiel:

```
' Liefert den Text des Eingabe Feldes und Checked/Unchecked der Checkbox.
Elo.CreateAutoDlg ("Personeninfo")
Elo.AddAutoDlgControl (4,1,"Name","")
Elo.AddAutoDlgControl (2,3,"Verheiratet? Ja","0")
Elo.ShowAutoDlg
MsgBox GetAutoDlgValue (1)
MsgBox GetAutoDlgValue (2)
```

Verfügbar ab: Ver 3.00.228

Siehe auch:

- CreateAutoDlg
- AddAutoDlgControl
- ShowAutoDlg

1.93 Funktion GetBarcode

Mit Hilfe dieser Funktion werden Barcodes, die zuvor mit Hilfe der Funktion ReadBarcodes ermittelt wurden ausgelesen.

```
AnsiString GetBarcode( int Index)
```

Parameter:

- Index Index des Barcodes

Rückgabewerte:

- <>"": Barcode
- "": Fehler (z.B. falscher Index, ReadBarcodes zuvor nicht aufgerufen)

Siehe auch:

- ReadBarcodes

1.94 Funktion GetChildNode

Mit dieser Funktion werden die Folgeknoten des aktuellen Knotens innerhalb eines Ablaufs in einer Schleife ausgelesen.

```
int GetChildNode()
```

Rückgabewerte:

- -2: keine weiteren Folgeknoten vorhanden
- -1: kein Knoten aktiv
- 1: Ok

Siehe auch:

- NodeActivateScript
- NodeAlertWait
- NodeAvailable
- NodeComment
- NodeFlowName
- NodeName
- NodeTerminateScript
- NodeType
- NodeUser
- NodeYesNoCondition
- NodeAllowActivate
- NodeObjectID
- OpenChildNodes
- SelectCurrentNode

1.95 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

1.96 Funktion GetCounter

Die Funktion GetCounter liefert den aktuellen Stand des ausgewählten Zählers. Wenn der Parameter Increment einen von 0 verschiedenen Wert enthält, wird der Zähler dabei automatisch hochgezählt.

```
AnsiString GetCounterList( AnsiString CounterName,  int Increment )
```

Parameter:

- CounterName Name des anzusprechenden Zählers
- Increment 0: nur aktuellen Zählerstand ermitteln,
- 1: Zählerstand ermitteln und hochzählen

Rückgabewerte:

- -1: Fehler
- ansonsten: Zählerstand.

Siehe auch:

- GetCounterList
- CreateCounter

1.97 Funktion GetCounterList

Die Funktion GetCounterList liefert einen String mit allen AccessManager Zählern, die augenblicklich im System angemeldet sind. Die einzelnen Einträge sind durch ein ¶ getrennt.

```
AnsiString GetCounterList()
```

Parameter:

- keine

Rückgabewerte:

- "": Fehler beim Lesen der Zählerliste
- ansonsten: Zählerliste.

Siehe auch:

- GetCounter
- CreateCounter

1.98 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.

Verfügbar seit: 3.00.392

Beispiel:

```
SET Elo=CreateObject("ELO.professional")
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

1.99 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:

- Aktuelle Dokumentenversion (Arbeitsversion) liefern

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

Verfügbar seit: 5.00.018

Beispiel:

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

1.100 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.

Verfügbar seit: 3.00.322

Beispiel:

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

1.101 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

Verfügbar seit: 5.00.018

Beispiel:

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

1.102 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

1.103 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

1.104 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.professional" )
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
```

1.105 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 "|".

Verfügbar seit: 3.00.322 für die Liste (Bit 3 des Status Feldes)

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

1.106 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.

Verfügbar seit: 5.00.018

Beispiel:

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

1.107 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

1.108 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

1.109 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

Verfügbar ab: 3.00.176

Siehe auch:

- ObjGuid
- GetObjFromGuid

1.110 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

Verfügbar seit: 3.00.170

Siehe auch:

- LookupHistMD5
- GetHistObj
- GetMD5Hash

1.111 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

Verfügbar seit: 3.00.170

Siehe auch:

- LookupHistMD5
- GetMD5Hash
- GetHistDoc

1.112 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

Verfügbar seit: 3.00.324

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.professional")
Liste=Elo.GetIndexGroups( "ELOFAQBER", "ELOVER", "%P3.%" )
Liste=Replace( Liste, "¶", vbcrlf )
MsgBox Liste
```

1.113 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

Verfügbar ab: 3.00.170

Siehe auch:

- Backup

1.114 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

Verfügbar seit: 3.00.170

Siehe auch:

- IntToDate

1.115 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

Verfügbar seit: 3.00.394

Beispiel:

```
Set Elo=CreateObject("ELO.professional")
UserId=Elo.ActiveUserId
' nur die eigenen Termine
'UserId=Elo.ActiveUserId + 1073741824 ' mit Gruppenterminen
'UserId=Elo.ActiveUserId + 536870912 ' mit Vertretungen
'UserId=Elo.ActiveUserId + 1610612736 ' alle Termine

cnt=Elo.CollectWv( UserId, "22.08.2002", "30.08.2002", 3)
for i=0 to cnt-1
  msg=msg & Elo.GetListEntry(i) & vbcrlf
next
MsgBox msg
```

Siehe auch:

- CollectWv
- DoInvisibleSearch

1.116 Funktion GetMaskLineAcl

Beim Bearbeiten einer Maskendefinition kann mit der Funktion GetMaskLineAcl die „Berechtigung“ einer Indexzeile abgefragt werden.

```
int GetMaskLineAcl( int iKeyNo)
```

Parameter:

- iKeyNo Indexzeilennummer (0 … 49)

Rückgabewerte:

- - : Leerstring bei unbekannter ‚iKeyNo‘
- ansonsten ACL der Indexzeile

Ab Client Version 8.00.056

Siehe auch:

- SetMaskLineAcl
- ReadObjMask
- WriteObjMask

1.117 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

Verfügbar: 3.00.170

Siehe auch:

- LookupHistMD5
- GetHistObj
- GetHistDoc

1.118 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

1.119 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

Verfügbar ab: 5.00.164

Siehe auch:

- GetObjAttrib
- GetObjAttribKey
- GetObjAttribName
- SetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

1.120 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

1.121 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

1.122 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

1.123 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

1.124 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: Aktenzeichen
- 4: ISO-Datum
- 5: Listeneintrag
- 6: Anwenderfeld
- 7: Thesaurus
- 8: Numerisch mit fester Breite
- 9: Numerisch mit fester Breite, 1 Nachkommastelle
- 10: Numerisch mit fester Breite, 2 Nachkommastelle
- 11: Numerisch mit fester Breite, 4 Nachkommastelle
- 12: Numerisch mit fester Breite, 6 Nachkommastelle

```
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

1.125 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

1.126 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:

- 0 Nur aktuelle Dokumentenversion (Arbeitsversion) berücksichtigen
- 1 Nur aktuelles Attachment berücksichtigen
- 2 Alle Dokumentenversionen berücksichtigen
- 3 Alle Attachmentversionen berücksichtigen
- 4 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

Verfügbar seit: 4.00.106

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.professional")

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

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

1.127 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
- Guid nicht gefunden
- >0 Objekt-Id

Verfügbar ab: 3.00.176

Siehe auch:

- GetGuidFromObj
- ObjGuid

1.128 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

1.129 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

Beispiel:

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

Siehe auch:

- InsertRef
- RemoveRef

1.130 Funktion GetOcrRectList

Diese Funktion gibt eine Liste von Trefferrechtecken zu einem gesuchten Wort zurück.

RectList enthält am Ende eine Liste der Trefferpositionen [xStart, yStart, xEnde, yEnde][…]

Wenn nach unterschiedlichen Begriffen gesucht werden soll, dann wird der eigentliche OCR Vorgang nur einmal ausgeführt, der Text wird im Client in einen internen Cache vorgehalten. Aus diesem Grund darf „FileName" auch nur einen gültigen ELO Archivdateinamen „12345678.TIF" enthalten. Eine Postboxdatei muss entsprechend umbenannt werden, die DocId sollte dabei so groß sein, dass sie sich nicht mit echten Dokumenten überkreuzt (z.B. ab 10000000). Der Treffercache wird bei jedem Clientneustart gelöscht.

```
AnsiString GetOCRRectList( AnsiString FileName, int PageNumber, AnsiString
SearchText, int Flags )
```

Parameter:

- FileName Name der zu durchsuchenden Datei. Es muss sich dabei um einen Namen im ELO Archivdateinamensformat handeln und dieser muss eindeutig sein.
- PageNumber Seitennummer des zu durchsuchenden Tiff Dokuments
- SearchText Gesuchter Begriff
- Flags Bit 0: Groß/Kleinschreibweise beachten

Rückgabewerte:

- Liste der Trefferpositionen [xStart, yStart, xEnde, yEnde][…]

Beispiel:

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

FileName = Elo.GetDocumentPath(Id, 0)

if FileName <> "" then
  RectList = Elo.GetOCRRectList(FileName, 1, "AccessManager", 0 )
  MsgBox RectList
else
  MsgBox "File not found"
end if
```

1.131 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

Verfügbar ab: 3.00.196

1.132 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

1.133 Funktion GetRegInfo

Mit dieser Funktion kann die Seriennummerninformation des AccessManagers abgefragt werden. Hierzu stehen über den Parameter Mode folgende Informationen zur Verfügung:

- 0: Name des Kunden auf den die Version Registriert ist
- 1: Anwenderanzahl im ELO Client
- 2: Anwenderantahl im Internet Gateway
- 3: Demoversion (TRUE/FALSE)
- 4: Startversion (TRUE/FALSE)
- 5: Volltextlizenz (TRUE/FALSE)
- 6: Replikationslizenz (TRUE/FALSE)
- 7: Backupserverlizenz (TRUE/FALSE)
- 8: Gültigkeitseinschränkung (Datum oder Leer)
- 9: Reiner Recherche-IGW (TRUE/FALSE)
- 10: Versendemappe (TRUE/FALSE)
- 11: Reine e-Mail Ablage (TRUE/FALSE)
- 20: Tobit Mailschnittstelle (TRUE/FALSE)
- 21: COLD (TRUE/FALSE)
- 22: XML Import (TRUE/FALSE)
- 23: SAPALINK Schnittstelle (TRUE/FALSE)
- 24: http DocServer (TRUE/FALSE)
- 25: Aktenzeichengenerator (TRUE/FALSE)
- 26: Signatur (TRUE/FALSE)
- 27: Stapelscan (TRUE/FALSE)

Alle Anfragen liefern einen Text zurück, mit der Anwenderanzahl, Anwendernamen (mehrzeilig) oder einem „TRUE“ oder „FALSE“ bei binären Anfragen.

```
AnsiString GetRegInfo(int Mode)
```

Parameter:

- Mode Auswahl der Seriennummerninformation 0..27

Rückgabewerte:

- Wert des Eintrags

Beispiel:

```
' GetRegInfo.VBS  28.01.2004
'-------------------------------------------------------------------
' © 2002 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo-digital.de)
'-------------------------------------------------------------------
' Dieses Skript liest die aktuelle Registrierungsinformation des
' AccessManagers aus. Die Funktion GetRegInfo erwartet dabei
' folgende Parameter:
' 0: Name des Kunden auf den die Version Registriert ist
' 1: Anwenderanzahl im ELO Client
' 2: Anwenderantahl im Internet Gateway
```

```
' 3: Demoversion (TRUE/FALSE)
' 4: Startversion (TRUE/FALSE)
' 5: Volltextlizenz (TRUE/FALSE)
' 6: Replikationslizenz (TRUE/FALSE)
' 7: Backupserverlizenz (TRUE/FALSE)
' 8: Datumseinschränkung
' 9: ReadOnly IGW
'10: Versendemappe
'11: Nur Mailablage
'12..19: Reserviert
'20: Tobit Schnittstelle
'21: COLD
'22: XML Importer
'23: SAPALINK Schnittstelle
'24: HTTP Doc Server
'25: Aktenzeichen
'26: Signatur
'27: Stapelscan
' ----------------------------------------------------------------

DIM Elo
DIM Text
DIM i
DIM Msg

Set Elo=CreateObject( "ELO.professional" )
Text=Array("Anwendername","Anwenderzahl","Internetanwender",_
           "DemoVersion","StartVersion","Volltextoption",_
           "Replikation","Backupserver","Datumsbegrenzung",_
           "RechercheIGW","Versendemappe","Nur e-Mail Ablage",_
           "Tobit Schnittstelle",_
           "COLD", "XML Importer", "SAPALINK", "Docserver",_
           "Aktenzeichengenerator", "Signatur", "Stapelscan")

for i=1 to 11
  Msg=Msg & Text(i) & " : " & Elo.GetRegInfo(i) & vbcrlf
next
for i=0 to 7
  Msg=Msg & Text(i+12) & " : " & Elo.GetRegInfo(i+20) & vbcrlf
next

Msg=Msg+vbcrlf
Msg=Msg & Text(0) & " : "  & vbcrlf & vbcrlf & Elo.GetRegInfo(0)

MsgBox Msg
```

Verfügbar seit: 3.00.282

1.134 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

1.135 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

Verfügbar ab: Ver 3.00.228

Beispiele:

- 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

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

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

In der Version 6.0 kann auch die ObjektIds Liste für die klassische Listenansicht zurück gegeben werden.

```
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

Verfügbar seit: 5.00.180

Beispiel:

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

1.137 Funktion GetUILanguage

Mit Hilfe dieser Funktion können Sie die aktuelle Spracheinstellung des ELO-Clients ermitteln.

```
AnsiString GetUILanguage
```

Rückgabewerte:

- „D“: deutsch
- „C“: tschechisch
- „E“: englisch
- „F“: französisch
- „I“: italienisch
- „K“: slovakisch
- „N“: niederländisch
- „P“: polnisch
- „S“: spanisch
- „DK“: dänisch

Verfügbar ab: 6.00.054

1.138 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

1.139 Funktion GotoPath

Über diese Funktion können Sie ELO veranlassen auf ein bestimmtes Dokument (oder Schrank, Ordner, Register) zu wechseln. Dazu können Sie hier den kompletten Zugriffspfad angeben und somit im Gegensatz zum Befehl GotoId auch Pfade über Referenzen beschreiben. Die Liste mit den ObjektIds muss mit dem normalen Trennzeichen gebildet werden und mit einer '1' beginnen.

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.

```
int GotoPath(AnsiString ObjektIds )
```

Parameter:

- ObjektIds Liste der ObjektIds, jeweils verbunden durch ein Trennzeichen

Rückgabewerte:

- -3: Pfad zum Objekt nicht gefunden
- -2: Ungültige ObjectId im Pfad
- -1: Kein Arbeitsbereich aktiv
- 1: ok

Verfügbar seit: 6.00.000

Beispiel:

```
Call Elo.GotoPath ("1¶123¶456¶789")
```

Siehe auch:

- GotoId
- GetTreePath

1.140 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
- 1 Ok

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

1.141 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
  Achtung: Parameter verursacht unter ELOprofessional und unter ELOoffice unterschiedliches Verhalten. Unter ELOprofessional verursacht der Wert ‚0‘ einen Fehler, unter ELOoffice gibt es bei einer ‚1‘ einen Fehler. (Problem ist bekannt, wird aber nicht mehr gefixt.)

Rückgabewerte:

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

Siehe auch:

- SetScriptMenu
- SetScriptButton
- GetScriptButton

1.142 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

1.143 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

Verfügbar ab: 3.00.176

Siehe auch:

- UpdateDocument
- GetDocumentPath

1.144 Funktion InsertProjectOptions

Über diese Funktion können Sie die Projektoptionen für die Aktivitätenliste ergänzen.

Zur Anmeldung eines neuen Projekt müssen Sie zuerst einen Eintrag unter dem Pseudo-Projekt "ELO_SYSTEM" mit der Major-Nummer 1 (Minor-Nummer können Sie auf 0 setzen, es wird dann automatisch die nächste freie Nummer vergeben) und dem Value Eintrag <Projektname> anlegen. Unter diesem Projektnamen können Sie dann die Listen für das Versandart-Feld, den Rückgabestatus, die Empfänger und die 10 Anwenderdefinierten Felder hinterlegen.

Eine Liste besteht immer aus einer Überschrift mit der Minornummer 1 und danach aus beliebig vielen Optionen zum Feldwert. Wenn die Überschrift mit einem !-Zeichen eingeleitet wird, kann der Anwender nur aus der Liste auswählen. In allen anderen Fällen kann der Anwender auch einen eigenen Wert eintragen.

Wenn zu einem der 10 Anwenderdefinierten Felder keine Optionenliste definiert wurde, dann wird es in der Bearbeitungsmaske nicht angezeigt. Auch wenn Sie keine Vorgaben in Form einer Stichwortliste machen können, so müssen Sie zumindest die Überschrift eintragen (Minornummer 1).

```
int InsertProjectOptions( AnsiString Project, int Major, int Minor, AnsiString
Value )
```

Parameter:

- Project Projektname für welches die Listen angelegt werden.
- Major 10 Empfängerliste
- 11 Versandart
- 12 Dokumentenstatus bei der Rückgabe
- 30..39 Anwenderdefiniertes Feld 1..10
- Minor Fortlaufende Nummer 1: Überschrift, 2: Option 1, 3: Option 2...
- Value Anzeigewert

Rückgabewerte:

- -2: Projektname fehlt
- -1: kein aktiver Arbeitsbereich vorhanden
- 1: Ok

Beispiel:

```
' Zuerst werden bereits vorhandene Einträge entfernt, das
' Skript trägt immer einen kompletten Projekdatensatz ein
call Elo.DeleteProjectOptions( "ELO" )

' Als nächstes wird das neue Projekt "ELO" angemeldet
call Elo.InsertProjectOptions( "ELO_SYSTEM", 1, 0, "ELO" )

' Es wird die Empfängerliste (Major-Nr. 10) definiert. Der
' Anwender kann nur einen Eintrag aus dieser Liste wählen,
' eigene Eingaben sind nicht möglich
call Elo.InsertProjectOptions( "ELO", 10, 1, "!Empfänger" )
```

```
call Elo.InsertProjectOptions( "ELO", 10, 7, "m.thiele" )
call Elo.InsertProjectOptions( "ELO", 10, 2, "g.schuster" )
call Elo.InsertProjectOptions( "ELO", 10, 3, "m.palkoska" )
call Elo.InsertProjectOptions( "ELO", 10, 4, "w.imig" )
call Elo.InsertProjectOptions( "ELO", 10, 5, "m.filkorn" )
call Elo.InsertProjectOptions( "ELO", 10, 6, "c.gembalski" )

' Die Versandartliste ist ebenfalls ein vordefiniertes
' Feld ohne eigene Eingabemöglichkeiten
call Elo.InsertProjectOptions( "ELO", 11, 1, "!Versandart" )
call Elo.InsertProjectOptions( "ELO", 11, 2, "Zur Ansicht" )
call Elo.InsertProjectOptions( "ELO", 11, 3, "Zur Freigabe" )
call Elo.InsertProjectOptions( "ELO", 11, 4, "Zur Bearbeitung" )

' Empfangsstatus
call Elo.InsertProjectOptions( "ELO", 12, 1, "!Status" )
call Elo.InsertProjectOptions( "ELO", 12, 2, "Freigegeben" )
call Elo.InsertProjectOptions( "ELO", 12, 3, "Bearbeitet" )
call Elo.InsertProjectOptions( "ELO", 12, 4, "Unverändert" )

' Erstes Anwenderdefiniertes Feld, Name "Produkt", Auswahl
' ausschließlich aus der Vorgabeliste
call Elo.InsertProjectOptions( "ELO", 30, 1, "!Produkt" )
call Elo.InsertProjectOptions( "ELO", 30, 2, "ELOprofessional" )
call Elo.InsertProjectOptions( "ELO", 30, 3, "ELOoffice" )
call Elo.InsertProjectOptions( "ELO", 30, 4, "ELOviewer" )

' Ein weiteres Anwenderdefiniertes Feld, der Anwender kann
' aus der Liste auswählen oder einen eigenen Wert eintragen
call Elo.InsertProjectOptions( "ELO", 33, 1, "Submodul" )
call Elo.InsertProjectOptions( "ELO", 33, 0, "Internet Gateway" )
call Elo.InsertProjectOptions( "ELO", 33, 0, "SAP Link" )
call Elo.InsertProjectOptions( "ELO", 33, 0, "Backup Server" )
call Elo.InsertProjectOptions( "ELO", 33, 0, "Mobil" )

' Noch ein Feld, keine Vorgabeliste
call Elo.InsertProjectOptions( "ELO", 34, 1, "Zusatzwunsch" )
```

Verfügbar ab: 3.00.360

Siehe auch:

- DeleteProjectOptions
- EditActivity
- ReadActivity
- WriteActivity
- NextActivity

1.145 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

1.146 Funktion InsertVTRep (int, int, AnsiString)

Über diese Funktion können Sie einen eigenen Text für die Volltextverschlagwortung hinterlegen.

```
int InsertVTRep( int ObjId, int Flag, AnsiString FileName )
```

Parameter

- ObjId: Dokument zu dem die Volltextdatei hinterlegt werden soll
- Flag: Bit 0: Flag „In den Volltext aufnehmen“ setzen.
- Bit 1: Vorhandene Volltextinformation überschreiben
- FileName: Name der Textdatei die die Volltextinformation enthält (*.TXT).

Rückgabewerte:

- -1: Kein Arbeitsbereich aktiv
- -2: Fehler beim Lesen der Datenbankinformation
- -3: VT-Datei bereits vorhanden
- -4: Fehler beim Kopieren der Datei
- -5: Fehler bei der Übergabe der Datei an den AccessManager
- -6, -7: Zielpfad konnte nicht ermittelt werden.
- >0: ok, Datei-Id

Verfügbar: 3.00.170

1.147 Funktion IntToDate

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

```
AnsiString IntToDate( int Datum )
```

Parameter:

- Datum Zu wandelnder Datumswert.

Rückgabewerte:

- Datumstext

Verfügbar ab 3.00.196

Siehe auch:

- DateToInt

1.148 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

1.149 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

1.150 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=Freigabe mit Zeitstempeleintrag,
- 1=Sperre mit Zeitstempeleintrag,
- 3=Sperre abfragen,
- 8=Freigabe ohne Zeitstempeleintrag,
- 9=Sperre ohne Zeitstempeleintrag

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.

1.151 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

1.152 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.

Verfügbar ab: 3.00.170

Siehe auch:

- LookupIndex

1.153 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

Verfügbar ab:4.00.034

Beispiel:

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

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
```

1.154 Funktion LookupHistMD5Ext

1.155 Funktion LookupHistMD5 (veraltet)

Diese Funktion ermittelt die Anzahl von Dokumenten, die einen vorgegebenen MD5 Hash Wert besitzen. Dieser Wert wird nur dann für neu abgelegte Dokumente ermittelt, wenn im AccessManager der Schalter CheckSumIn auf true gesetzt wurde. Über den Mode Parameter können Sie durch die Bit Position 0 bestimmen, ob gelöschte Dokument unterdrückt werden sollen.

```
int LookupHistMD5Ext( AnsiString MD5, int Mode )
int LookupHistMD5( AnsiString MD5 )
```

Parameter:

- MD5 zu suchender MD5 Hash Wert (als 32 Byte Hex-Zeichenfolge).
- Mode 0: gelöschte Dokumente anzeigen
- 1: gelöschte Dokumente unterdrücken

Rückgabewerte:

- -1 kein Arbeitsbereich aktiv
- 0..n Anzahl der Dokumente mit diesem Hash Wert

Verfügbar: 3.00.170 und3.00.332 (Ext Version)

Siehe auch:

- GetMD5Hash
- GetHistObj
- GetHistDoc

1.156 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

1.157 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

1.158 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

1.159 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.professional")
Name=InputBox("Bitte einen Anwender- oder Gruppennamen eingeben")
MsgBox Elo.LookupUserName(Name)
```

Siehe auch:

- LookupKeyName
- ReadUser

1.160 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 Ablagemaske verwendbar
  - 1 als Ablagemaske verwendbar
- Bit 4:
  - 0 nicht als Recherchemaske verwendbar
  - 1 als Recherchemaske 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

1.161 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
- DocKind
- DocPath

1.162 Funktion MergeImages / 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.professional")
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
```

1.163 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

1.164 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

1.165 Funktion MovePostboxFile / 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

Verfügbar ab: (MovePostbox2) 3.00.276, (MovePostbox)3.00.216

Siehe auch:

- ActivePostFile

1.166 Funktion MoveToArchive / 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:

- -4: Dokument konnte nicht ins Archiv verschoben werden

- -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

1.167 Funktion MsgBox

Anzeige einer Standard Messagebox.

```
MsgBox(String Nachricht, String Titel, int Buttons)
```

Parameter:

- Entsprechen denen der Microsoft Standard Messagebox

Verfügbar seit: 7.00.052

Beispiel:

```
Call Elo.MsgBox ("Nachricht", "Titel", 0)
```

1.168 Property NoteOwner(int)

Dieses Property enthält beim Skript-Event Aufruf "Beim Bearbeiten einer Haftnotiz" den Eigentümer der aktiven Haftnotiz. Das Skript Event wird vor dem Bearbeiten-Dialog und nach dem Bearbeiten Dialog aktiviert. Vor dem Dialog steht der ScriptActionKey auf 1, nach dem Bearbeiten auf 2 bei Ok und 3 bei Abbruch.

Verfügbar seit: 3.00.330; 5.00.224

Siehe auch:

- NoteText
- NoteType
- NoteAcl

1.169 Property NoteText (AnsiString)

Dieses Property enthält beim Skript-Event Aufruf "Beim Bearbeiten einer Haftnotiz" den Text der aktiven Haftnotiz. Das Skript Event wird vor dem Bearbeiten-Dialog und nach dem Bearbeiten Dialog aktiviert. Vor dem Dialog steht der ScriptActionKey auf 1, nach dem Bearbeiten auf 2 bei Ok und 3 bei Abbruch.

Verfügbar seit: 3.00.330; 5.00.224

Beispiel:

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

Siehe auch:

- NoteType
- NoteOwner
- NoteAcl

1.170 Property NoteType(int)

Dieses Property enthält beim Skript-Event Aufruf "Beim Bearbeiten einer Haftnotiz" den Typ (Haftnotiz, persönliche Haftnotiz oder Stempel) der aktiven Haftnotiz. Das Skript Event wird vor dem Bearbeiten-Dialog und nach dem Bearbeiten Dialog aktiviert. Vor dem Dialog steht der ScriptActionKey auf 1, nach dem Bearbeiten auf 2 bei Ok und 3 bei Abbruch.

Verfügbar seit: 3.00.330; 5.00.224

Siehe auch:

- NoteText
- NoteOwner

1.171 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
- L Ein Anwender- oder Gruppeneintrag mit Recht "Listen bearb."

Die Kennzeichen R, W, D, E, L 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..

Verfügbar: 4.00.000

Beispiel:

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

1.172 Property ObjBarcodeInfo(AnsiString)

Über das Property ObjBarcodeInfo können Sie Barcode Konfiguration für den ausgewählten Dokumententyp abfragen. Dieser Text wird über die Ablagemaskenverwaltung im Feld BarcodeInfo eingetragen und der internen Barcodeanalyse zur Verfügung gestellt. Sie können diesen Text aber auch extern auswerten, z.B. um eigene Barcode Komponenten einzufügen.

Beispiel:

```
set Elo = CreateObject("ELO.professional")
MaskNo=Elo.LookupMaskName( "Barcodemaske" )
x=Elo.PrepareObjectEx(0,254,MaskNo)
MsgBox "BarcodeInfo: " & Elo.ObjBarcodeInfo
```

Verfügbar: 3.00.264

1.173 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

#define MFG_ISREPLROOT 0x80000 // Replikationskreis Startknoten

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.
- Bit 29: 1 Bei einer Suche wird die Versionsgeschichte mit durchsucht

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

Beispiel:

Auslesen der Informationen:

```
If (Elo.ObjFlags And 1)= 1 Then ' Versionskontrolle an
If (Elo.ObjFlags And 2)= 2 Then ' Revisionssicher abgelegt
If (Elo.ObjFlags And 64)= 64 Then ' Dokument im Volltext
If (Elo.ObjFlags And 256)=256 Then ' Dokument verschlüsselt
```

Setzen der Informationen:

```
Elo.ObjFlags=Elo.ObjFlags Or 1 ' Versionskontrolle an
Elo.ObjFlags=Elo.ObjFlags Or 2 ' Revisionssicher abgelegt
Elo.ObjFlags=Elo.ObjFlags Or 64 ' Dokument im Volltext
Elo.ObjFlags=Elo.ObjFlags Or 256 ' Dokument verschlüsselt
```

Löschen der Informationen:

```
Elo.ObjFlags=Elo.ObjFlags XOr 1 ' Versionskontrolle an
Elo.ObjFlags=Elo.ObjFlags XOr 2 ' Revisionssicher abgelegt
Elo.ObjFlags=Elo.ObjFlags XOr 64 ' Dokument im Volltext
Elo.ObjFlags=Elo.ObjFlags XOr 256 ' Dokument verschlüsselt
```

Siehe auch:

- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- MaskFlags

1.174 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

Verfügbar: 3.00.180

Siehe auch:

- GetGuidFromObj
- GetObjFromGuid

1.175 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

1.176 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

1.177 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

1.178 Property ObjKey (int) (invalid)

Das Property ObjKey setzt oder liest den Schlüssel eines Eintrags.

Siehe auch:

- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- ObjSReg

1.179 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.

Verfügbar ab: 3.00.188

Beispiel:

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

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

1.180 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

1.181 Property ObjMaskNo (int)

Das Property ObjMaskNo setzt oder liest den aktuell eingestellten Dokumententyp.

Siehe auch:

- ObjFlags

1.182 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

1.183 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

1.184 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

1.185 Property ObjOwner (int)

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

Siehe auch:

- ObjInfo

1.186 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

1.187 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

1.188 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

1.189 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

1.190 Property ObjType (int) (invalid)

1.191 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

1.192 Property ObjVDate (AnsiString)

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

1.193 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

1.194 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

1.195 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

1.196 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

1.197 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 Fehlerfall ein Leerstring)

1.198 Funktion OcrGetText

Nach der OCR Analyse stehen in einem Textfeld die erkannten Teile der Rechecckliste 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).

1.199 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.

1.200 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.

1.201 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.

Rückgabewerte:

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

1.202 Property (AnsiString) OpenSave (int Wert)

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

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

Optionen:

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.

Verfügbar ab: Ver 3.00.228

Beispiel:

Setzen der Default Dateierweitung auf EXE

```
ELO. OpenSave (1) = „EXE“
```

Einlesen des Übergebenen Dateinamens

```
Filename = ELO. OpenSave (3)
```

Siehe auch:

- OpenSaveDialog

1.203 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)

Beispiel:

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.

Verfügbar ab: Ver 3.00.228

Siehe auch:

- OpenSave

1.204 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

1.205 Property OutlookName (AnsiString)

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

Siehe auch:

- DelOutlookName

1.206 Property PopupObjID

Wird die Funktion "ClickOn" mit einer Funktion aus dem Kontextmenü verwendet, muss dem Property PopupObjID die Objekt-ID des zu bearbeitenden ELO-Eintrags zugewiesen werden.

Beispiel:

```
Elo.PopupObjID = Elo.GetEntryID(-1)
```

1.207 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

1.208 Funktion PrepareObject (invalid)

1.209 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.

Soll nach dem PrepareObjectEx eine Suche mit DoSearch oder DoSearchEx durchgeführt werden, übergeben Sie als ObjectId den Wert –3 (ab Client-Version 6.00.054).

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,
- -2: aktueller Eintrag wird als ‚Neu' markiert
- -3: Eintrag für nachfolgende Suche initialisieren
- 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.professional")

' 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

1.210 Funktion (int) PrintDocList

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
-
- 0 – Listendarstellung
- 1 – Spaltendarstellung
- 0 – Kurze Überschriften
- 1 – Lange Überschriften
- 0 – Keine Trennlinie
- 1 – Trennlinie nach jedem Eintrag
- 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)

Beispiel:

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)
```

Verfügbar ab: Ver 3.00.228

Siehe auch:

- PrintDocListTabs

1.211 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 voreingestellten Werte.

Folgende Eingaben sind zulässig:

- 1 Setzt Portrait Default Werte
- 2 Setzt Landscape Default Werte

Verfügbar ab: Ver 3.00.228

Beispiel:

Setzt die Defaultwerte für Portrait Druck.

```
ELO. PrintDocListTabs (0) = 1
```

Siehe auch:

- PrintDocList

1.212 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
-

1.213 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

Verfügbar: 3.00.204

Siehe auch:

- ObjAcl

1.214 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.professional" )

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
```

Verfügbar seit: 3.00.286

Siehe auch:

- SetOption

1.215 Funktion ReadBarcodes

Diese Funktion liest Barcodes, die sich auf einem Postbox-TIFF-Dokument befinden. 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. Die Funktion liefert die Anzahl der erkannten Barcodes zurück, die Barcodes können mit Hilfe der Funktion *GetBarcode* abgerufen werden.

Der Parameter BarcodeDescriptor enthält die Liste der zu untersuchenden Rechtecke. Eine ausführliche Beschreibung dieser Liste finden Sie in der Barcode-Dokumentation.

```
int ReadBarcodes( AnsiString SourceFile, AnsiString BarcodeDescriptor, int
MaskNo, int ReadMultiple )
```

Parameter:

- SourceFile Zu untersuchende Datei oder Postlisten-Zeilennummer (mit #Nummer).
- BarcodeDescriptor Rechteckliste, im Barcode-Format.
- MaskNo Dokumententyp der zu erzeugenden Schlagwortdatei
- ReadMultiple 1 = alle Barcodes im definierten Rechteck lesen
- 0 = nur einen Barcode lesen

Rückgabewerte:

- >0: Anzahl der erkannten Barcodes
- -1: kein Workspace offen
- -2: keine Rechteckliste vorhanden
- -3: fehlerhafte Rechteckliste
- -4: Datei existiert nicht
- -5: Barcode-Komponente konnte nicht initialisiert werden
- -6: Ungültiges TIFF-Format

Siehe auch:

- GetBarcode
- AnalyzeFile

1.216 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

1.217 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.

Beispiel:

Gibt den Namen des Schlüssels mit der Nummer 1 aus.

```
MsgBox Elo.ReadKey (1)
```

Siehe auch:

- WriteKey

1.218 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

1.219 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

Verfügbar ab 5.00.066

Beispiel

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

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", ".", " - " )
...
```

1.220 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
- 3 Der angemeldete User ist nicht der Administrator des eingelesenen Users und die UserFlags wurden auf 0 gesetzt.

Siehe auch:

- ReadUser
- UserGroups
- UserParent
- UserKeys
- UserFlags

1.221 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
' Archivnamen, gefolgt von einem Archivpfad.
' ----------------------------------------------------------------

Set Elo=CreateObject("ELO.professional")

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

1.222 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

1.223 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

1.224 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

Verfügbar seit: 5.00.094

Beispiel:

```
Elo.RemoveDoc (1, "20050727", 3)
```

1.225 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

Beispiel:

- Entfernt die fünfte Referenz von Objekt mit der Id 34.
- Elo.RemoveRef (34,5)

Siehe auch:

- InsertRef
- GetObjRef

1.226 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

1.227 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

1.228 Funktion SaveDocumentPage

Mit dieser Funktion kann eine Seite einer Multipage-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

1.229 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

1.230 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

1.231 Property ScriptActionKey (int)

Über dieses Property werden Informationen vom Script an ELO zurückgeliefert. ELO reagiert darauf mit ganz bestimmten Aktionen.

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

1.232 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

Verfügbar seit: 5.00.036

Beispiel:

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

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

MsgBox res
```

Siehe auch:

- UnselectSearchListLine
- SelectSearchListLine
- SearchListLineSelected

1.233 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

1.234 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

1.235 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.professional" )

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

1.236 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

1.237 Funktion SelectPostBoxLine / 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

1.238 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

1.239 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

Verfügbar: 4.00.152

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.professional")

Elo.SelectView 4
Elo.PrepareObjectEx 0, 0, 2
Elo.SetObjAttrib 0, "%Thiele%"
Elo.DoSearch
Elo.SelectTreePath "Lager¶Einkauf¶Lagereingang¶*"
```

Siehe auch:

- SelectView
- SelectTreePathEx

1.240 Funktion SelectTreePathEx

Dieser Befehl aktiviert in der Suchansicht den TreeView mit einer virtuellen Ansicht und blättert ihn gemäß des 1. Parameterstrings auf. (Siehe Beschreibung zu SelectTreePath)

Über den 2. Parameter wird die virtuelle Ansicht definiert. Gibt man hier ein '*' ein, wird in diesem Fall der klassische Ordner-Tree angezeigt.

```
int SelectTreePathEx( AnsiString TreePath, AnsiString DefAnsicht)
```

Parameter:

- TreePath Aufzublätternder Pfad im TreeView
- DefAnsicht Definition für eine virtuelle Ansicht

Rückgabewerte:

- -1: Kein Arbeitsbereich aktiv
- 1: Ok

Verfügbar: 7.00.032

Beispiel:

Emails werden nach Absender und Empfänger strukturiert:

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

call Elo.SelectTreePathEx("", "MailVonAn|2:1,0,1,-1,-1,-1")
```

Der Text mit der Definition beginnt mit dem Namen „MailVonAn“. Danach folgt ein Pipe-Symbol als Trenner, die Nummer der Ablagemaske („2“) und ein Doppelpunkt als weiterer Trenner mit der nachfolgenden Information, ob leere Indexeinträge aufgefüllt werden sollen (0 oder 1 im Beispiel „1“). Durch Komma getrennt folgen nun die Nummern der Indexzeilen. Es müssen immer genau 5 Einträge sein, nicht verwendete Indexzeilen werden durch ein „-1“ gekennzeichnet.

Siehe auch:

- SelectView
- SelectTreePath

1.241 Funktion SelectUser / 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

Verfügbar: (Ex) 3.00.276

Beispiel:

```
Set Elo=CreateObject("ELO.professional")
Users = Elo.SelectUserEx(1)
UserList = split( Users, "," )
for each UserNo in UserList
  MsgBox Elo.LoadUserName(UserNo)
Next
```

Siehe auch:

- FindUser
- LoadUserName
- ActiveUserId

1.242 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:

- ViewId0: 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

1.243 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.

```
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

Beispiel:

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

1.244 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

1.245 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.

Verfügbar seit: 4.00.212

Beispiel:

```
set Elo = CreateObject("ELO.professional")
file=Elo.ActivePostFile
if file <> "" then
  MsgBox Elo.SeparateTiffFile( file, -1, 4, 0 )
end if
```

1.246 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

1.247 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

1.248 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

Verfügbar ab: 5.00.164

Siehe auch:

- SetObjAttrib
- SetObjAttribKey
- SetObjAttribName
- GetObjAttribFlags
- SetObjAttribMin
- SetObjAttribMax
- SetObjAttribType

1.249 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

1.250 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

1.251 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

1.252 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

1.253 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: Aktenzeichen
- 4: ISO-Datum
- 5: Listeneintrag
- 6: Anwenderfeld
- 7: Thesaurus
- 8: Numerisch mit fester Breite
- 9: Numerisch mit fester Breite, 1 Nachkommastelle
- 10: Numerisch mit fester Breite, 2 Nachkommastelle
- 11: Numerisch mit fester Breite, 4 Nachkommastelle
- 12: Numerisch mit fester Breite, 6 Nachkommastelle

```
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

1.254 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“

Verfügbar seit: 3.00.286; Option No 3 und später: ab 3.00.350

Beispiel

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

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

1.255 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 12 Buttons)
- 3: Postbox (bis zu 12 Buttons)
- 4: Recherche (bis zu 12 Buttons)
- 5: Wiedervorlage (bis zu 12 Buttons)
- iButton Nummer des Buttons (1..16 bzw. 1..12)
- sScriptName Name des Skripts (ohne Extension)
- iVisible 1 = Button sichtbar schalten
- 0 = Button unsichtbar schalten

Ab ELO Version 8 wird die Multifunktionsleiste eingesetzt.

Die Ribbontab-Nummern für die Funktion SetScriptButton und die Namen für die Funktion AddRibbonGroup.

- 1: sRibbonTabName="dxRibbon1Start";
- 2: sRibbonTabName="dxRibbon1Dokument";
- 3: sRibbonTabName="dxRibbon1Archiv";
- 4: sRibbonTabName="dxRibbon1Aufgaben";
- 5: sRibbonTabName="dxRibbon1Ansicht";
- 6: sRibbonTabName="dxRibbon1Zwischenablage";
- 7: sRibbonTabName="dxRibbon1Postbox2"; //Postbox ablegen
- 8: sRibbonTabName="dxRibbon1Suche";
- 9: sRibbonTabName="dxRibbon1Postbox"; // Postbox Scannen
- 10: sRibbonTabName="dxRibbon1AufgabenTools";

Der 2. Funktionsparameter wird dann für die Angabe der Gruppe innerhalb des ‚Tabs' verwendet.

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

1.256 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

Beispiel:

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

Zusätzlich Verfügbar ab der Version 4.00.190:

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
sEdeleteEntry	33	Beim Löschen eines Eintrags.

Siehe auch:

- GetScriptEvent

1.257 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

Verfügbar ab 5.00.064

1.258 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

Zum Entfernen eines Skriptes aus dem Kontextmenü wird einfach eine ‚1000' auf den Wert des Parameters ‚TabSheet' addiert. Also z.B.: Elo.SetScriptMenü 1001, „Script A"

Siehe auch:

- SetScriptButton
- GetScriptButton
- ImportScript

1.259 Funktion SetViewersReadOnly

Mit dieser Funktion kann die Annotationleiste bei Anzeige eines TIFF Dokumentes auf ‚ReadOnly' gesetzt werden. Damit können vom Anwender keine Notizen mehr auf dem Dokument angebracht bzw. bearbeitet werden.

Das Setzen der Option erfordert einen Viewer Refresh.

```
int SetViewersReadOnly(int ReadOnly)
```

Parameter:

- ReadOnly
- 1: Bearbeitungsmöglichkeit von Notizen abschalten
- 2: Bearbeitungsmöglichkeit von Notizen einschalten
-

Rückgabewerte:

- -1: Kein Arbeitsbereich aktiv
- 1: OK

1.260 Funktion ShowAutoDlg ()

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.

Verfügbar ab: Ver 3.00.228

Beispiel:

Erstellt einen leeren Dialog und zeigt ihn an.

```
Elo.CreateAutoDlg („Mein Dialog“)
Elo.ShowAutoDlg
```

Siehe auch:

- CreateAutoDlg
- AddAutoDlgControl
- GetAutoDlgValue

1.261 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

1.262 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.

Verfügbar seit: 5.00.042

Siehe auch:

- MaskKey
- DocKey
- DocKind
- DocPath

1.263 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

Verfügbar seit: 4.00.054

1.264 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

Verfügbar 3.00.220

1.265 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

1.266 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

1.267 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

1.268 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

1.269 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

1.270 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.

1.271 Funktion ToClipboard

Diese Funktion übergibt einen Text an das Windows Clipboard

```
Int ToClipboard( AnsiString Text )
```

Parameter:

- Text Zu übergebender Text

Rückgabewerte:

- Immer o

Verfügbar seit: 3.00.456

Beispiel:

```
Option Explicit

Dim Elo,url,id,Guid,iObjID,iZeile,Body

set Elo=CreateObject("ELO.professional")
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

1.272 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

Verfügbar ab: 3.00.324 (Professional); 5.00.222 (Office)

Beispiel, Skript 1 mit dem Namen "Callback"

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

1.273 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

1.274 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.professional" )

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

1.275 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

1.276 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

1.277 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

1.278 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

Verfügbar ab Ex-Version ab 3.00.170

Siehe auch:

- InsertDocAttachment
- GetDocumentPath

1.279 Funktion UpdateNote

Die Funktion aktualisiert eine Haftnotiz, auf die zuvor mit FindFirstNote oder FindNextNote zugegriffen wurde.

```
int UpdateNote()
```

Parameter:

- -

Rückgabewerte:

- 1: Haftnotiz wurde aktualisiert
- -1: kein Arbeitsbereich aktiv
- -2: keine Haftnotiz selektiert
- -3: Fehler beim Datenbankupdate
- -4: keine ausreichenden Rechte zum Schreiben von Haftnotizen vorhanden
- -5: die zu aktualisierende Haftnotiz ist ein Stempel
- -6: die zu aktualisierende Haftnotiz wird gerade von einem anderen User bearbeitet

Verfügbar seit: 6.00.090

Beispiel:

```
' Text der ersten Haftnotiz eines Dokuments mit der
' Objekt-ID iID ergänzen:
Set Elo=CreateObject("ELO.professional")
Elo.PrepareObjectEx iID,0,0
iRes=Elo.FindFirstNote
If iRes=1 Then
  Elo.NoteText=Elo.NoteText & vbCRLF & Now
  IRes=Elo.UpdateNote
End If
```

Siehe auch:

- FindFirstNote
- FindNextNote
- DeleteNote

1.280 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

1.281 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

1.282 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

Addiert man auf die beschriebenen Werte noch eine ‚16', wird intern zusätzlich ein ‚ProcessMessages' ausgeführt. Dies führt zum Abarbeiten der Windows Message Queue

Rückgabewerte:

- -1: Kein Arbeitsbereich aktiv
- >=0: Anzahl der Einträge in der Postbox (bei Sammeln) oder 1

Siehe auch:

- UpdateObject
- UpdateDocument

1.283 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

Verfügbar ab 5.00.066

Beispiel

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

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", ".", " - " )
...
```

1.284 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
- UserFlags2

1.285 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

1.286 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

1.287 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

1.288 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

1.289 Property UserTerminate (AnsiString)

Das Property enthält den Namen des Anwenders, der diesen Knoten terminiert hat. Der Name wird beim Abschluß des Knotens in Textform gespeichert. Er bleibt also auch dann erhalten, wenn der originale Anwender im ELO AccessManager gelöscht oder umbenannt wird.

Verfügbar seit: 3.00.508.

Siehe auch:

- NodeAction
- NodeActivateScript
- NodeAlertWait
- NodeAvailable
- NodeComment
- NodeFlowName
- NodeTerminateScript
- NodeType
- NodeUser
- NodeYesNoCondition

1.290 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.

Achtung: Ab ELO 8 gibt es eine ‚verlängerte' Versionsnummer: Beispiel: 8000068000

```
int Version()
```

Parameter:

- keine

Rückgabewerte:

- Elo-Versionsnummer

1.291 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.

1.292 Property WindowState (int)

Über dieses Property können Sie den Zustand des ELO Fensters abfragen.

- Werte: Normal : 1
- Minimiert: 2
- Maximiert: 3
- Unbekannt: -1

Verfügbar: 3.00.218

1.293 Funktion WriteActivity (AnsiString)

Diese Funktion schreibt einen Aktivitäteneintrag. Wenn das Feld ActGuid leer ist, wird ein neuer Eintrag erzeugt, wenn hier ein Wert eingetragen ist, erfolgt ein Update auf den ausgewählten Datensatz.

Der Parameter-String setzt sich aus folgenden Teilen zusammen:

- ActGuid Eindeutige Kennung der Aktivität, bei einem Neueintrag leer lassen!
- DocGuid GUID des ELO-Dokuments (per GetGuidFromObj ermitteln).
- Destination Empfänger
- RevVers Version
- ActTStamp Zeitstempel, leer lassen, wird beim Speichern automatisch erzeugt
- Project Projektname, nach Möglichkeit vorbelegen
- Owner Eigentümer, ELO Anwendernummer
- Creator Erzeuger, ELO Anwendernummer
- Prio 0,1 oder 2
- ShortDesc Kurzbezeichnung
- SentAt Versendedatum ISO-Format
- SentMode Versandart
- DueDate Erwartetes Rückgabedatum, ISO-Format
- BackAt Rückgabedatum, ISO-Format
- BackMode Rückgabestatus
- Comment Zusatztext
- FileName Versendedateiname
- UD0 Anwenderdefiniertes Feld 1
- UD1
- …
- UD9 Anwenderdefiniertes Feld 10

```
int WriteActivity( AnsiString Activity )
```

Parameter:

- Activity: Zu schreibender Datensatz

Rückgabewert:

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

Verfügbar seit: 3.00.360

Beispiel:

```
Id=Elo.GetEntryId(-1)
if (Id>1) then
  guid=Elo.GetGuidFromObj(Id)
```

```
  res="¶" & guid
  res=Elo.EditActivity( res )
  MsgBox res
  Elo.WriteActivity( res )
end if
```

Siehe auch:

- ShowActivities
- EditActivity
- ReadActivity
- WriteActivity
- NextActivity

1.294 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

1.295 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

Beispiel:

Erstellt einen neuen Systemschlüssel Nummer 10 mit dem Namen Buchhaltung.

```
Elo.WriteKey (10,"Buchhaltung")
```

Siehe auch:

- ReadKey

1.296 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: (läuft ab Version 3.00.290)

```
' Anlegen einer neuen Maske mit einer Indexzeile
NEWMASK=9999
Set Elo=CreateObject("ELO.professional")

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
- SetMaskLineAcl

1.297 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

1.298 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.professional")
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.professional")
if Elo.ReadUser( Elo.ActiveUserId )>0 then
  StartObj=Elo.ReadUserProperty(5)
  if StartObj<>"" then
    Elo.GotoId(StartObj)
  end if
end if
```

Verfügbar seit: zusätzliche Option zum direkten Speichern 3.00.358

Siehe auch:

- ReadUserProperty
- UserGroups
- UserParent
- UserKeys
- UserFlags

1.299 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

1.300 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

1.301 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

1.302 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

1.303 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

1.304 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

1.305 Funktion WvListInvalidate ()

Diese Funktion aktualisiert die Wiedervorlageliste.

```
int WvListInvalidate ()
```

Rückgabewerte:

- 1 Aktualisierung erfolgreich
- -1 kein Arbeitsbereich aktiv

Siehe auch:

- WvNew
- EditWv
- WvDueDate
- WvActionCode

1.306 Property WvNew (int)

Dieses Property gibt an ob eine neue Wiedervorlage erstellt wurde.

Werte:

- 0 Nein
- 1 Ja

Siehe auch:

- EditWV
- WvDueDate
- WvListInvalidate
- WvActionCode

1.307 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

1.308 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

1.309 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

1.310 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

1.311 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

1.312 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

1.313 Anhang A

Bezeichnung der Dialogelemente in der ELO Hauptansicht bis Version 7.0

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, „...Popup...“: Menüpunkt innerhalb eines Kontext-Menüs, „Plp...“: Menüpunkt innerhalb des Kontext-Menüs der Postbox, „...UserBtn...“: Skript-Buttons.

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:Zur Position der letzten Ablage springen	BtGotoArc
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: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:Workflow annehmen	wvTakeFlow
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	ClZoomSel
B:Zoom 25%	ClZoom25
B:Zoom 50%	ClZoom50
B:Zoom 100%	ClZoom100
B:Breite einpassen	ClZoomWidth
B:Dokument einpassen	ClZoomFit
B:90° nach links drehen	ClRot270
B:Auf den Kopf stellen	ClRot180
B:90° nach rechts drehen	ClRot90
B:Anwenderdefinierte Vergrößerung	ClZoomUser
B:Aktuelles Dokument drucken	ClPrint
B:Fax	ClFax
B:Aktuelles Dokument durchsuchen	BtCheckSearch
B:Dokumentensperre erneuern	mbNewCheckOut
B: Zur letzten Dokumentenanzeige zurück	mbDocBack
B:Zur nächsten Dokumentenanzeige vor	mbDocFore
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: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: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:Zurückstellung aufheben	WorkflowResetDelay
M:Report	mmWFRep
M:Übersicht	Report2
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: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: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:Konvertierung nach PDF	PlpConvPDF
M:Vertretungs-Postboxen einsehen	mmShowVertIntray
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: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:Report...	Report1
M:Standardregister einfügen...	ArcPopupInsertStdReg
M:Volltext...	mnVolltext1
M:Volltextinhalt abrufen (erstellt Postboxdatei ..	mnFultextInfo
M:Wiedervorlagetermine...	ArcPopupWv
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:Wiedervorlagetermin anlegen...	ArcPopupToWv

M:Löschen...	ArcPopupRemove
M:Hilfe...	Hilfe4
M:Gehe zu	WvGotoEntry
M:Termin bearbeiten...	WvEditEntry
M:Termin löschen	WvDelEntry
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:Annahme verweigern	KomPlpRefuse
M:Gehe zu	cpGoto
M:Einchecken	cpCheckIn
M:Bearbeiten	cpEdit
M:Drucken	cpPrint

M:Verwerfen	cbDel

1.314 Anhang B

Bezeichnung der Aktionen in der ELO Hauptansicht ab Version 8.0

Ab der Version 8.0 wird programmintern mit Aktionen gearbeitet. Für die Überlagerung von Aktionen mit Skripten oder deren Aufruf per „ClickOn“, muss ab dieser Version mit den Aktionsnamen anstelle der in Anhang A genannten Controlnamen gearbeitet werden. Eine Liste der Aktionsnamen finden Sie im nachfolgend eingebetten PDF-Dokument, die Aktivierung des Dokuments erfolgt per Doppelklick.

	ACT_PrintDocList	Liste drucken
	ACT_VersionCompare	Versionsvergleich
	ACT_BindToArchiveDoc	Neue Version
	ACT_ForwardOneStep	Einen Schritt vor
	ACT_ScanAddToArchiveDoc	Seite(n) an Archivdokument anfügen
	ACT_SendPDF	Versenden als PDF
	ACT_SearchNote	Notizen suchen
	ACT_SearchVersionComments	Versionskommentare durchsuchen
	ACT_SearchFullText	Volltextsuche

	ACT_AMDiag	Angemeldete Anwender
	ACT_ServerStatus	Serverstatus
	ACT_RepOptions	Reportoptionen
	ACT_OpenSubTree	Teilbaum komplett öffnen
	ACT_ListACLs	Berechtigungen auflisten
	ACT_EditIndex_Popup	Verschlagwortung
	ACT_EditACL_Popup	Berechtigungen setzen
	ACT_CountFolder_Popup	Archiveinträge zählen
	ACT_ShowDocument_Popup	Zur Ansicht öffnen
	ACT_CheckOutDoc_Popup	Auschecken und bearbeiten
	ACT_CheckInDoc_Popup	Einchecken

	ACT_ConvPDF_Popup	PDF-Konvertierung
	ACT_ConvTIFF_Popup	TIFF Konvertierung
	ACT_NewFolder_Popup	Neuer Ordner
	ACT_ClipboardCopy_Popup	Kopieren
	ACT_ClipboardInsert_Popup	Referenz erstellen
	ACT_ClipboardMove_Popup	Eintrag verschieben
	ACT_PrintDocument_Popup	Dokument drucken
	ACT_SaveDocument_popup	Datei speichern unter
	ACT_WFStart_Popup	Workflow starten
	ACT_WFAdhoc_Popup	Ad-hoc-Workflow
	ACT_WFEntryFlowsOwnActive_Popup	Eigene aktive Workflows

	ACT_WFEntriesClosed_Popup	Abgeschlossene Workflows
	ACT_NewWv_Popup	Wiedervorlage
	ACT_ShowWvs_Popup	Wiedervorlagen zum Eintrag
	ACT_Link_Popup	Verlinkung
	ACT_Delete_Popup	Löschen
	ACT_BackToRoot	Archivanfang
	ACT_BackOneStep	Einen Schritt zurück
	ACT_InsertDocLeft	Dokument aus Vorlage
	ACT_NewFolder	Neuer Ordner
	ACT_EditIndex	Verschlagwortung
	ACT_CheckOutDoc	Auschecken und bearbeiten

	ACT_CheckinDoc	Einchecken
	ACT_CollectOutlook	Outlook
	ACT_SendEMail	Dokument versenden
	ACT_PrintDocument	Dokument drucken
	ACT_QuickInfo	Ouickinfo anzeigen
	ACT_InsertCopy	Einfügen
	ACT_FullScreen	Vollbildansicht
	ACT_Clipboard	Auf das Klemmbrett legen
	ACT_ShowDocument	Zur Ansicht öffnen
	ACT_DocumentVersions	Dokument Versionen
	ACT_GotoEntry	Gehe zu

	ACT_EditDocument	Dokument bearbeiten
	ACT_SendLink	Versenden als Link
	ACT_ClipboardCopy	Kopieren
	ACT_ClipboardInsert	Referenz erstellen
	ACT_ClipboardMove	Eintrag verschieben
	ACT_Delete	Löschen
	ACT_DeleteFromClipboard	Vom Klemmbrett entfernen
	ACT_DeleteFromSearchList	Aus dem Suchergebnis entfernen
	ACT_StartSearch	Suche starten
	ACT_Refresh	Aktualisieren
	ACT_ChangePassword	Passwort ändern...

	ACT_Configuration	Systemeinstellungen...
	ACT_Terminate	ELO beenden
	ACT_Help	Hilfe
	ACT_LoadNewDocument	Neue Version laden
	ACT_AddPages	Scanseiten anfügen
	ACT_CreatePreview	Vorschau-Dokument erstellen
	ACT_SaveDocument	Datei speichern unter
	ACT_CreateSignature	Signatur erstellen
	ACT_CheckSignature	Signatur prüfen
	ACT_Checksum	Checksumme prüfen
	ACT_OpenAttachment	Dateianbindung zur Ansicht öffnen

	ACT_AttachmentVersions	Versionen der Dateianbindung
	ACT_AddAttachment	Dateianbindung hinzufügen
	ACT_SaveAttachment	Dateianbindung speichern unter
	ACT_DeleteAttachment	Dateianbindung löschen
	ACT_Link	Verlinkung
	ACT_ShowReport	Report zum Archiv
	ACT_CountFolder	Archiveinträge zählen
	ACT_EditACL	Berechtigungen setzen
	ACT_AddRegisterTemplate	Standardregister einfügen
	ACT_SaveRegisterTemplate	Als Standardregister speichern
	ACT_AddDocumentFile	Datei einfügen

	ACT_Export	Export
	ACT_Import	Import
	ACT_Unlock	Sperre entfernen
	ACT_AddToFulltext	In Volltext aufnehmen
	ACT_DeleteFromFulltext	Aus Volltext entfernen
	ACT_ShowDeleted	Gelöschte Einträge einblenden
	ACT_Undelete	Wiederherstellen
	ACT_DieHard	Gelöschte Einträge dauerhaft entfernen
	ACT_SearchOptions	Weitere Optionen
	ACT_ExtendedSearch	Verschlagwortung durchsuchen
	ACT_WvPrioA	Priorität A

	ACT_WvPrioB	Priorität B
	ACT_WvPrioC	Priorität C
	ACT_NewWv	Wiedervorlage
	ACT_EditWv	Aufgabe ändern
	ACT_DeleteWv	Wiedervorlage löschen
	ACT_ScanSingle	Seiten scannen
	ACT_ScanMultiple	Dokument scannen
	ACT_CollapsePages	Seiten klammern
	ACT_CollapseSeparationSheet	Klammern (Trennseiten)
	ACT_SeparatePages	Seiten trennen
	ACT_PostboxInsertPage	Datei einfügen

	ACT_PostboxEditIndex	Verschlagwortung
	ACT_PostboxDeleteIndex	Verschlagwortung löschen
	ACT_Barcode	Barcode- Erkennung
	ACT_PostboxDelete	Löschen
	ACT_MoveToArchive	Archivablage
	ACT_MoveToArchiveAuto	Automatische Ablage
	ACT_AddPagesPostbox	Seiten anfügen
	ACT_EditScanProfiles	Scan-Profile
	ACT_SelectScannerPostbox	Scanner auswählen
	ACT_OptUser	Anwender
	ACT_OptMasks	Verschlagwortungsmasken

	ACT_OptDocumentPaths	Dokumentenpfade
	ACT_OptColorConfig	Schriftfarbe
	ACT_OptReport	Report einschalten
	ACT_OptKeys	Schlüssel
	ACT_OptProjects	Aktivitätenprojekte
	ACT_OptScripts	Skripte
	ACT_OptOptions	Konfiguration...
	ACT_AboutELO	Über das Programm...
	ACT_FaxDocument	Dokument faxen
	ACT_Thumbnails	Miniaturansicht
	ACT_DoubleView	Visueller Vergleich

	ACT_ReScan	Seite erneut scannen
	ACT_SendFromPostbox	Verschieben in andere Postbox
	ACT_ShowSearchTree	Baumansicht anzeigen
	ACT_PrepareDVDOutput	Lesekopie des Archivs
	ACT_WVReport	Übersicht Wiedervorlagen
	ACT_ConvPDF	PDF-Konvertierung
	ACT_ConvTIFF	TIFF Konvertierung
	ACT_TwinFree	Freie Anordnung
	ACT_TwinSideBySide	Nebeneinander anordnen
	ACT_TwinOverUnder	Übereinander anordnen
	ACT_SlideShow	Diashow starten

	ACT_MarkAll	Alles markieren
	ACT_ListToSearch	Liste auf die Suchansicht legen
	ACT_DocToPostbox	Kopie in Postbox
	ACT_NavigateToTemplateFolder	Vorlagen
	ACT_ShowTIFFPreview	Vorschaudokument anzeigen
	ACT_BuzzwordsGlobal	Stichwortliste Global
	ACT_BuzzwordsVersionComment	Stichwortliste Versionskommentar
	ACT_BuzzwordsVersionNumbers	Stichwortliste Versionsnummern
	ACT_Cancel	Abbruch
	ACT_SysInfoCenter	System Infocenter
	ACT_SystemDiag	Systemdiagnose

	ACT_PrintArchive	Archivübersicht drucken
	ALT_CheckDocManager	Archivinhalte prüfen
	ACT_DelVersions	Gelöschte Versionen entfernen
	ACT_DelOldDocs	Altdokumente entfernen
	ACT_DelVerfallsDocs	Verfallsdokumente löschen
	ACT_OnlineFAQ	Online FAQ im Internet
	ACT_ScanToArchive	Ins Archiv scannen
	ACT_MultiColumnSearchList	Mehrspaltige Anzeige
	ACT_VoiceInfo	Sprachnotiz
	ACT_FullTextContent	Volltextinhalt anzeigen
	ACT_RotateSort	Drehen/ Sortieren

	ACT_CopyToOtherPostbox	Kopie in andere Postbox
	Act_SaveAnonym	Direktablage anonym
	ACT_CrankPages	Seiten verschränken
	ACT_ListExcel	Listenausgabe Excel
	ACT_ELOConnector	Connector
	ACT_NewActivity	Aktivität
	ACT_DailyRet	Tagesrückgaben
	ACT_DeadLines	Überschrittene Termine
	ACT_ShowWvs	Wiedervorlagen zum Eintrag
	ACT_RegisterActivity	Zur Überwachung anmelden
	ACT_ReturnDocument	Rückgabe

	ACT_DocumentActivities	Aktivitäten zum Eintrag
	ACT_BindOutlook	Ordner einblenden
	ACT_SyncOutlook	Synchronisieren
	ACT_ToOutlook	Dokumente übertragen
	ACT_EditCryptParams	Verschlüsselungskreise
	ACT_MovoToSelectedArchiveReg	Direktablage
	ACT_CollectOutlook2	Outlook-Ordner archivieren
	ACT_CheckUpdate	Updateprüfung
	ACT_ConfigConnector	Connector konfigurieren
	ACT_RandNotiz1	Neue Randnotiz
	ACT_RandNotiz2	Persönliche Randnotiz

	ACT_RandNotiz3	Permanente Randnotiz
	ACT_RandNotiz1a	Allgemeine Randnotiz
	ACT_ReportOptions	Reportoptionen
	ACT_DiscardChanges	Dokumentenänderungen verwerfen
	ACT_PreprocessPostbox	OCR Vorverarbeitung
	ACT_MoveDocuments	Dokumentdateien verschieben
	ACT_ShowEntryReport	Report zum Eintrag
	ACT_UserFeedback	Nutzer-Feedback
	ACT_ConfSubstitutes	Vertretungsregelung
	ACT_BuzzwordsWorkflow	Stichwortliste Workflow
	ACT_TranslationMap	Übersetzungstabelle

	ACT_WorkflowTemplates	Workflow-Vorlagen
	ACT_ConfAZ	Aktenplan
	ACT_EditReplZones	Replikationskreise
	ACT_AMKey	Archivschlüssel
	ACT_CheckACLs	Berechtigungen prüfen
	ACT_CheckCharSet	Zeichensatzkontrolle
	ACT_CycleCheck	Zyklusprüfung
	ACT_WFCancel	Zurückstellung löschen
	ACT_WFDefer	Workflow zurückstellen
	ACT_WFDelegate	Workflow delegieren
	ACT_WFStart	Workflow starten

	ACT_WFActiveShow	Übersicht Workflows
	ACT_WFAccept	Workflow annehmen
	ACT_WFConfirm	Workflow weiterleiten
	ACT_WFRelease	Workflow abgeben
	ACT_WFAdhoc	Ad-hoc- Workflow
	ACT_WFEdit	Workflow anzeigen
	ACT_WFEntryFlowsActive	Aktive Workflows
	ACT_WFReject	Workflow zurückgeben
	ACT_WFSetSubstitute	Vertreter einsetzen
	ACT_NewsMsg	Neue Nachricht
	ACT_FilterReplSets	Replikationskreis- Filter

	ACT_WFEntryFlowsOwnActive	Eigene aktive Workflows
	ACT_WFEntriesClosed	Abgeschlossene Workflows
	ACT_WorkflowsActiveDialog	Workflows zur Bearbeitung
	ACT_WFEscalation	Workflow Fristüberschreitungen
	ACT_ShowVertPostbox	Vertretungs-Postboxen einsehen
	ACT_AssignReplSets	Replikationskreise zuordnen
	ACT_TasksToExcel	Listenausgabe Excel
	ACT_TasksToHTML	Listenausgabe HTML
	ACT_WFEntriesAllActive	Aktive Workflows anzeigen
	ACT_Backup	Backup
	ACT_FTService	Volltextdienst

	ACT_SrcToExcel	Listenausgabe Excel
	ACT_SrcToHTML	Listenausgabe HTML
	ACT_SearchBin	Gelöschte Einträge suchen
	ACT_BatchEdit	Stapel- Verschlagwortung
	ACT_HorizontalSplit	ACT_HorizontalSplit
	ACT_VerticalSplit	ACT_VerticalSplit
	ACT_WvSortorder	Sortierreihenfolge
	ACT_WvCollapseExpand	Erweitern/reduzieren
	ACT_iSDate	Datum
	ACT_iSMask	Maske
	ACT_iSType	Objekttyp

	ACT_iSOwner	Abgelegt von
	ACT_iSAddField	Indexfeld
	ACT_Undo	Rückgängig