ELOoffice OLE Automation Interface

[Date: 2019-01-08 | Program version : 10.05.000]

Note: The description of the ELOoffice OLE Automation Interface is an excerpt from the ELOprofessional OLE Automation documentation. As the two interfaces are almost identical, a large number of scripts can be modified with little effort. However, calls to functions that are only available for ELOprofessional or ELOenterprise are not available for the ELOoffice OLE Automation Interface. Individual functions may be described in this documentation although they are not available in the ELOoffice OLE Automation Interface. If you find any such references, please inform our support team. You can reach them at support@elo.com. This version of the document is based on the ELOprofessional OLE Automation Interface version dated June 24, 2015.

Structure and application of the OLE Automation Interface

The ELOoffice 2.1 DDE Interface has been replaced by a more modern and improved OLE Automation Interface in the professional version. It is now possible to send commands to ELO externally, make entries, or query information from ELO.

As a rule, script names can be assigned freely. They must only satisfy the rules for file names. All script names starting with "ELO" are reserved for ELO Digital Office. Do not use script names that start with ELO to prevent naming conflicts.

The ELO OLE Automation Interface is closely based on the earlier DDE. For this reason, it should be thought of as command-oriented rather than object-oriented.

The interface is more or less the same for ELOoffice and ELOprofessional. However, not all functions are available for ELOoffice. When creating objects, note that ELOoffice uses a different identifier than ELOprofessional. In ELOoffice, the object must be created with ELO.office as the identifier instead of ELO.professional.

Important Information:

Please note the main features of the OLE Automation Interface:

- Add functions to the ELO client quickly and easily

- Maximum upwards compatibility between versions

Both of these goals mean that new automation commands may be introduced, sometimes as the result of projects that are only designed to perform a specific function. As a consequence, it cannot be guaranteed that every command works properly in every work area and in combination with any other commands. Due to the compatibility requirement, new client functions cannot always be introduced to the existing commands. In such a case, either an xyz_EXT function is created, or the functionality is not exported via the automation interface at all.

Keep in mind that the interface is a client interface. It was not designed to perform a large number of actions. If you plan to carry out a large number of actions, you should first check whether the OLE interface can meet your requirements. As many components are involved here which we do not have direct control over, we are unable to provide a workaround for some restrictions (such as memory leaks in the OLE interface, in the Windows Script Host, or even in the compiler runtime environment).

Contents

1.1 Starting ELO, selecting a repository, and the logon dialog box

There are two basic scenarios for controlling ELO remotely:

- A user is working with ELO and you want to use your application to perform actions in their workspace. In this case, it is not necessary to log on, as the user is already logged on. Your application only needs to check whether ELO is already running.
- You have an independent process (e.g., automatic fax or mail transfer). In this case, the application must log on (and log off at the end).

To log on to ELO in the first scenario:

```
// Start ELOprofessional ...
try
  {
    EloServer= CreateOleObject("ELO.professional");
  }
catch (...)
  {
    // ELOprofessional must be started at least once
    // at each workstation - it then registers
    // automatically as an OLE Automation Server
    return;
  };

// You should first make sure that a user
// is logged on.
ObjId=EloServer.OleFunction("GetEntryId",-1);
if (ObjId==-1)
  {
    // Error: no active work area, i.e., ELO is
    // running, but no user is logged on
    return;
  };
```

To log on to ELO in the second scenario:

```
// ELOprofessional starten ...
try
  {
    EloServer= CreateOleObject("ELO.professional");
  }
catch (...)
  {
    // ELOprofessional must be started at least once
    // at each workstation - it then registers
    // automatically as an OLE Automation Server
    return;
  };

// check version
iOleResult=EloServer.OleFunction("Version");
if (iOleResult<101002)
```

```
  {
    // Version is older than 1.01.002
    // too old!
    EloServer.OleFunction("Login","LOGOUT","","");
    EloServer=NULL;
    return;
  };

// System logon
iOleResult=EloServer.OleFunction("Login", Loginname,
                                 Passwort, Archiv );
if (iOleResult<0)
  {
    // Unknown logon name, password, or repository
    EloServer.OleFunction("Login","LOGOUT","","");
    EloServer=NULL;
  };
```

To log off from ELO in the second scenario:

```
// Log off from the repository and close ELO again
EloServer.OleFunction("Login","LOGOUT","","");
EloServer=NULL;
```

1.1.1 Using the separator "¶"

The separator "¶" (ALT 0182) is used to specify paths. This used to be "¿" (ALT 0191).

To also take future changes into account, we advise developers to use the "LookupDelimiter" function. This function returns the currently valid separator.

1.1.2 Identifying the currently selected entry

With the GetEntryId function, you can query the currently selected entry. GetEntryName then returns the corresponding name.

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

1.1.3 List of all documents in a folder

First of all, the ObjectId of the child folder 1998 (contained in the Invoices folder, under the Accounting top-level folder) is determined with LookupIndex. After switching to the folder, all documents can be queried.

```
// First, find the folder
RegisterId=EloServer.OleFunction("LookupIndex",
                                 "¶Buchhaltung¶Rechnung¶1998");
if (RegisterId>0)
  {
    // now call up the folder you have found
    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;

        // the document is added to a list here
      };
  };
```

1.1.4 Adding a new document to the folder

In order to add a new document, first you have to prepare an empty entry, enter keywording, and transfer the image file to the user's Intray with AddToPostbox. This entry can then be transferred to the repository.

```
// 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);

// send it to the Intray
iOleResult=EloServer.OleFunction("AddPostboxFile",
                                 "c:\temp\ford.tif");

// and transfer it from the Intray to the repository
sDestInfo="¶Urlaub 98¶Florida Westküste¶Ford Meyers";
iOleResult=EloServer.OleFunction("MoveToArchive",sDestInfo);
```

1.1.5 Detecting the keywording form (document type) number from the name

If you want to add a document to the repository, you must first determine the document type. ELO works with keywording form numbers; however the rest of the world uses names. In order to convert the name into a keywording form number, you have to browse through the available keywording forms and compare them with the name:

```
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;
  };
```

However, it is easier to use the LookupMaskName function, available in version 1.01.048 and higher. The function described above is combined into one call:

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

1.1.6 Adding an e-mail with message body and file attachment to the repository

The following example shows how to automatically add an e-mail to the repository. First, the e-mail is read, the subject, sender, and recipient are evaluated, and the body and attachments are saved. The basic information and file name are transferred to the InsertIntoELO function.

```
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);
 // version-controlled
    if (iMailMask>0)
      {
        // the EloMail keywording form has two
        // additional entries: sender and recipient
        EloServer.OleFunction("SetObjAttrib",0,sFrom);
        EloServer.OleFunction("SetObjAttrib",1,sTo);
      };

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

    // if no repository target is available,
    // the e-mail remains in the Intray
    if (iOleResult>0 && sDestination!="")
      {
        iOleResult=EloServer.OleFunction("MoveToArchive",
                                         sDestination);

        // the new ObjectId is identified here ...
        iObjId=EloServer.OleFunction("GetEntryId",-2);
        if (iOleResult>0)
          {
            // ... and the file is attached.
            iOleResult=EloServer.OleFunction("InsertDocAttachment",
                                             iObjId,sMailAttachment);
```

```
        };
    };

  return iOleResult>0;
};
```

1.1.7 Editing a keywording form

You can read and save document type information with the ReadObjMask and WriteObjMask commands. The following values are available as **document type information**:

ObjMName	The keywording form name
ObjMIndex	The target information for automatic filing
MaskFlags	Bit ...
MaskKey	Keywording form key
DocKey	Default value for document key
DocKind	Default value for document color
DocPath	Document filing path
DocTPath	Default value for the filing path

In addition, there are **50 attribute lines** (0 to 49) available. Each attribute line has the following values:

Get/SetObjAttrib()	User input, such as an invoice number
Get/SetObjAttribName()	Name of a keywording form field, e.g. "Invoice number"
Get/SetObjAttribKey()	Index name in the database, such as "INVNO"
Get/SetObjAttribType()	0: Text, 1: Date, 2: Numeric
Get/SetObjAttribMin()	Minimum input length in digits, 0: No minimum
Get/SetObjAttribMax()	Maximum input length in digits, 0: No maximum

The ObjAttrib field is not relevant when defining a keywording form. It is not saved with the other information, as it is only completed when users add documents to the repository.

A differentiation is made between the ObjAttribName and ObjAttribKey attributes for three reasons:

- If different language versions need to be used in parallel, keywording forms can be provided with corresponding versions (such as "Invoice number" and "Rechnungsnummer" for the same field). Internally, the database uses the same key for both cases (such as "INVNO").

- A keywording form can contain multiple fields of the same type. For example, a "Book" keywording form may contain multiple fields for the author (author, co-author, and so on). All of these fields can then be configured in the database with the same internal key (such as AUTHOR) so that they are all considered equally in a search.
- Different keywording forms may also contain fields of the same type. An invoice could have the "Customer name" field, while a delivery note might have the "Supplier name" field. Both fields can possess the same internal database key - "NAME" - and both be triggered in a search.

Note that the minimum and maximum input refer to the number of characters, not the entered value (including numeric fields).

The index fields 0-49 are available as keywording form fields. Next, there are ten fields that are reserved for internal ELO tasks. Field 50 currently contains link information. The index field is assigned the ELO_XLINK group name in the AttribName field, and a random 12-digit string in the Field value.

Here is a brief example showing how to create a new keywording form:

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

if Elo.ReadObjMask( NEWMASK )<0 then
  MsgBox "Error preparing keywording form"
else
  Elo.ObjMName="ELO test keywording form"
  Elo.MaskFlags=25

  ' An index field, input length 5..10 characters
  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 "Error no. " & x & " while creating new keywording form."
  else
    MsgBox "New keywording form with number " & Elo.ObjMaskNo & " created."
  end if
end if
```

1.1.8 Reading a color value or the entire color table

To read color values, use the ReadColorInfo function and the ColorInfo and ColorName properties.

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

A special form of ReadColorInfo is available to identify the complete color table. If you set bit 0x8000 in the color number, the function reads the desired color or, if there is none, the next highest color number.

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

    MyColorNumber= EloServer.OlePropertyGet("ColorNo");
    // Now edit the current color value here

    MyColorNumber++;
  };
```

1.1.9 Querying and configuring the work area

The SelectView function is available for querying or configuring the current work area. When you call this function with parameter 0, the number of the currently selected work area is returned (1-5). If you use a value of 1-5, the display switches to the chosen work area.

```
// Request current status
ActView=EloServer.OleFunction("SelectView",0);

if (ActView!=3)
  {
    // is not set to the Intray -> switch
    EloServer.OleFunction("SelectView",3);
  };
```

1.1.10 Edit Intray contents

The following example shows how an external program or an ELO scripting macro can search the Intray and process the available entries.

In this example, all entries are loaded, then the text "Important" is added to the "Extra text" field for all selected entries, while "Unimportant" is added for all non-selected entries.

```
for (iPostLine=0;i<1000;i++)  // process a maximum of 1000 entries
  {
    // first, load the desired line
    iRes=EloServer.OleFunction("PrepareObject",-1,iPostLine,0)
    switch (iRes)
      {
        case -5: continue;    // no keywording available
    case -7: continue;   // no keywording available
        case -6: return true; // finished, no further entries
        case 3:  sText="Important"; break;
        case 4:  sText="Unimportant"; break;
        default: return false; // something went wrong
      };
```

```
    // Read content of the extra text field, add additional text and return
    sText=EloServer.OlePropertyGet("ObjMemo")+sText;
    EloServer.OlePropertySet("ObjMemo",sText);

    // save modified data set
    EloServer.OleFunction("AddPostboxFile","");
  };
```

1.1.11 Scanning documents in landscape format

Many sheetfed scanners can only scan A4 documents in portrait format. If a number of documents need to be scanned in landscape, this requires the user to take action for each document. The process can be automated with a simple script. Create a new script with the following content:

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

Now register this script for the "after scanning" event. Each page is automatically rotated when scanned.

This script can also easily be extended to search the complete list of documents in the Intray for selected entries and rotate them. In this form, it can be placed on a custom script button, allowing larger groups of documents to be rotated at once.

1.1.12 Finding entries via the OLE interface

This example is based on the following scenario:

In a company, delivery notes are printed with barcodes. Employees in the warehouse add handwritten notes to these documents, so they cannot be directly transferred via COLD. Instead, they are scanned again and filed to the repository via the barcode component.

In the example, the order management department needs to regularly check the ELO documents to see if the delivery notes have been scanned, adding information as needed.

The keywording form for delivery notes contains a field for "Delivery note number – DLNO". The barcode component enters the data to this field. The data for the "Customer number – CTNO" and "Delivery date – DLDAT" fields are added by the order management department.

To prepare the subsequent fields, the application must generate an EloServer object and identify a keywording form on the forms list with the name "Delivery notes" (iLfMaskNo).

The order management department must first go through all incomplete delivery notes. For each delivery note, the following routine is called (please note that a "real" routine must check for errors. This part has been removed here in the interest of clarity):

```
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: // error
          return false;
        case 0:  // not found
          return false;
        case 1:  // found – enter now
          break;
        default: // Multiple entries - this must be dealt with somehow
                 // e.g. all entries are expanded.
          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;
  }
```

As a result of the function call, the order management department only needs to output an error message or note in their internal database that the comparison was performed successfully. This method has the advantage of checking whether all delivery notes have been scanned, so the data that already exists in the system (customer number, date) does not need to be entered again.

1.1.13 Example: Outputting an access path to the currently selected object

```
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 "Access path: " &STemp
```

1.1.14 Example: Blank and separator page check in the Intray

```
' TestPage.VBS  10.10.2001
'------------------------------------------------------------' © 2001 ELO
Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo-digital.de)
'------------------------------------------------------------' This script
checks all entries in the Intray for blank
' or separator pages. If a blank page (at least 97% white content
' in each segment) is found, the script also checks how much
' content is blank and returns the corresponding
' value.
```

```
' ------------------------------------------------------------
set Elo = CreateObject("ELO.professional")
ver=Elo.Version
if ver<300220 then
  MsgBox "This script requires ELOprofessional 3.00.220 or higher"
else
  ' Run through all Intray entries (maximum of 100 entries)
  for i=0 to 100
    ' Eintrag in den Viewer auswählen
    if Elo.SelectLine(i)<0 then
      exit for
    end if

    ' Check for separator and blank pages at the same time
    res=Elo.CheckPage(3, 970)
    if res=1 or res=3 then
      ' in the case of a blank page, determine the maximum value
      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

    ' Add separator text
    if res=2 or res=3 then sTmp=sTmp & " [ IsTrennseite ] "

    ' Add file name to notification
    sPathName=Elo.ActivePostFile
    sName=Elo.SplitFileName( sPathName, 2 )
    sTmp=sTmp & " : " &sName & vbcrlf

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

    ' Takes the SelectLine back
    Elo.UnselectPostboxLine(i)
  next

  ' Display result in a MessageBox.
  MsgBox sTmp
End if
```

1.1.15 Simple form management

The following example goes through the entries in the Intray, takes them, and uses OCR software to search a defined area for the keywords "Invoice" and "Delivery note". If one of these keywords is found, the script loads the corresponding keywording form definition and searches additional areas of the document for invoice number, customer number, etc. These entries are then saved along with the document and filed to the repository.

The program reads all selected entries in the Intray and

- ' checks whether the document belongs to the "Invoice" or "Delivery note"
- ' form type. The corresponding keywording form field data is
- ' read from the document and saved.
- ' Finally, the document is moved to the repository.

```
set ELO = CreateObject("ELO.professional")
iNum=0
Do while (1)
  iRet=Elo.PrepareObject (-1,iNum,0)
  If (iRet = -6) Then
    MsgBox "Program finished!"
    Exit Do
  End If


  if (iRet = 3 Or iRet = -7) Then   ' Only process selected documents.
    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,"Invoice")>0) then
        res=Elo.Status("Invoice in row " & strRow & " detected.")
        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=Invoice"
          iRet=Elo.MoveToArchive (ObjIndex)
          if (iRet <> 1) then
            msgbox "Error during transfer, RetValue = " & Cstr(iRet)
          End If
        End If
      Else
        If (InStr(Elo.ObjShort,"Delivery note")>0) then
          res=Elo.Status("Delivery note in row " & strRow & " detected.")
          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 "Error during transfer, RetValue = " & Cstr(iRet)
          End If
        Else
          res=Elo.Status( "Unknown document in " & strRow )
          If SaveObjShort="" then
            Elo.ObjShort="Unknown document"
          Else
            Elo.ObjShort=SaveObjShort
          End If
          iRet=Elo.AddPostBoxFile("")
        End If
      End If
    Else
      res=Elo.Status("Document type could not be checked in row " & strRow)
      If SaveObjShort="" then
            Elo.ObjShort="Unknown document"
      Else
        Elo.ObjShort=SaveObjShort
      End If
      iRet=Elo.AddPostBoxFile("")
    End If
  End If
iNum=iNum+1
Loop

res=Elo.UpdatePostbox()
```

1.1.16 Example: Before editing a sticky note

When editing a sticky note, the name and time are automatically entered into the text field. The ScriptActionKey parameter contains values 1: before being called, 2: after being called and ended with Ok, and 3: after being called and ended with Cancel.

```
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 Example of a Microsoft Winword 8 macro

The following lines contain an example for automatically transferring a document from WinWord to ELO using a macro.

```
Public Sub MAIN()
TempVerz$ = Environ("TEMP")
' Temporary directory from environment variables
If Documents.Count < 1 Then          ' If no document is open
    MsgBox ("Transfer not possible because no documents are open")
    GoTo ENDE
End If
If Tasks.Exists("The electronic Leitz folder") = False then
    ' if ELO  is not active ->
    GoTo StartEloProgram          ' Message that ELO must be started
End If
i = 1                            ' Find possible temp file name "WW_ELOi.doc"
Loop:                            ' that does not exist in window list
For j = 1 To Application.Windows.Count      '  Search through all opened windows
    If Application.Windows(j).Caption="WW_ELO"+LTrim(Str(i))+".doc" Then
       i = i + 1                     '  next temporary file name "WW_ELOi.doc"
       GoTo loop                  '  search window list
    End If
Next j
TempName$ = "WW_ELO" + LTrim(Str(i)) + ".doc"
If Dir(ActiveDocument.FullName) <> "" Then  ' Has the document already been
saved?
   If ActiveDocument.Saved = False then     ' Saved, but modified
MsgText$ = "Document was modified and has been assigned the temporary file name:
" + TempName$       + Chr$(13) + Chr$(10)
        Dateiname$ = TempVerz$ + "\" + TempName$
        ActiveDocument.SaveAs FileName:=Dateiname$
   Else                                     ' is saved and has not been modified
  MsgText$ = ""                   ' no indication that document only exists as
temp file
        FileName$ = ActiveDocument.FullName ' File name+path (copies original
file)
   End If
Else                             ' Document is a new document
   MsgText$ = "Warning: The document has not yet been saved and has been given
the temporary file name: " + TempName$       + Chr$(13) + Chr$(10)
   FileName$ = TempDir$ + "\" + TempName$    ' set temporary file name
   ActiveDocument.SaveAs FileName:=FileName$  ' save under temporary file name
End If
KurzName = ActiveDocument.BuiltInDocumentProperties("Title")
If ShortName = "" Then                ' if no title entered, it is assigned
  ShortName = ActiveDocument.Name    ' the document name as short name
End If
Kom = ActiveDocument.BuiltInDocumentProperties("Comments")   ' Übernahme des
Kommentars
DDate$ = ActiveDocument.BuiltInDocumentProperties("Last Save Time")
DDate$ = DateValue(DDate$)  ' Determines valid date: dd.mm.yy
Dim ELOServer As Object
Set ELOServer = CreateObject("ELO.professional") ' ELO32 OLE-Server
On Error GoTo LOGIN                          ' ELO is in logon dialog box
iOleResult = ELOServer.PrepareObject(0, 4, 0)     ' new entry, document, default
On Error GoTo 0
ELOServer.ObjMemo = Com              ' Document comment
```

```
ELOServer.ObjXDate = DDate$          ' last save date
EntryID = ELOServer.GetEntryID(-12)  ' >0 -> An activated directory exists
If (ELOServer.SelectView(0) = 1) And (EntryID > 0) Then ' ELO is in "Repository
+ Directory"
   txt = MsgText$ + Chr$(13) + Chr$(10) + "The document will be filed to the
repository with the following short name."
   ShortName = InputBox(txt, "ELOprofessional document transfer", ShortName)

                                     ' The short name can be changed via the
   ELOServer.ObjShort = ShortName     ' dialog box
   iOleResult = ELOServer.AddPostboxFile(FileName$)
' first to the Intray
   iOleResult = ELOServer.MoveToArchive("#" + Str(EntryID))
' then to the repository
Else
' otherwise transfer to the Intray
   txt = MsgText$ + Chr$(13) + Chr$(10) + "The document will be moved to the
Intray and given the following short name."
   ELOServer.ObjShort = InputBox(txt, "ELOprofessional document transfer",
ShortName)
   iOleResult = ELOServer.AddPostboxFile(File name$)       ' transfer to Intray
End If
Set ELOServer = Nothing         ' release server object again
GoTo ENDE

' LOGIN:
Set ELOServer = Nothing         ' release server object again
MsgBox ("Error: You must log on to ELOprofessional.")
GoTo ENDE

EloProgrammStarten:
MsgBox ("Error: ELOprofessional must be running for the transfer to be
successful.")

ENDE:
End Sub
```

1.1.18 Scripting in workflows

In workflows, two scripts can be assigned to each node, which are each automatically processed when entering and exiting the node. These scripts can access the data in the current node. The following properties are available:

NodeAction, NodeActivateScript, NodeAlertWait, NodeAvailable, NodeComment, NodeFlowName, NodeName, NodeTerminateScript, NodeType, NodeUser and NodeYesNoCondition (see below for a description of each of these properties).

Before these properties are accessed, the NodeAvailable property must be checked. If its value is 1, information about a node is available, otherwise no information is available. For example, if the script is run manually (i.e. not as part of a workflow), no information can be provided regarding a specific node. In this case, the NodeAvailable property has the value 0.

Caution: When forwarding a workflow, the entire workflow is read first, together with the ELO keywording information for the document. Next, the required steps in the workflow and the relevant scripts are executed. If you now use one of the scripts to change the keywording information, your changes will not yet be visible during this step. This can become critical for decision nodes especially. If the workflow takes a particular fork based on the content of the "AMOUNT" field, for example, and this field is changed in the end script of the node immediately preceding the decision node, the new value will not be taken into account by the decision node. However, if you need to perform an operation of this type, you can place an exclamation point in front of the field's group name in the decision node. This tells the workflow to read the corresponding field directly from the database and not from the internal data.

A sample script that accesses the data of a node:

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

  FlowTxt="Name of process: "&Elo.NodeFlowName

  Select Case Elo.NodeType
    Case 1  NTxt="Start node"
    Case 2  NTxt="Person"
    Case 3  NTxt="Distribution node"
    Case 4  NTxt="Decision node"
    Case 5  NTxt="Collection node"
  End Select

  if Elo.NodeAction=1 Then
    NTxt=NTxt&" enabled."
  Else
    NTxt=NTxt&" terminated."
  End If
  NTxt="Aktion: "&NTxt

  NameTxt="Node name: "&Elo.NodeName

  CommentTxt="Comment: "&Elo.NodeComment

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

  ActivateTxt="Script when activating: "&Elo.NodeActivateScript
  TerminateTxt="Script when terminating: "&Elo.NodeTerminateScript

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

End If
```

If the active node is a distribution node, you can specify the successor node for the document using a script. This script is entered to the *End script* field of a (distribution) node:

```
Set Elo=CreateObject("ELO.professional")
ObjID=Elo.NodeObjectID              'Access to the data...
Elo.PrepareObject ObjID,0,0         '...of the document being processed
MsgBox Elo.ObjShort
Rv=Elo.OpenChildNodes               'Access to the successor nodes
If Rv=1 Then
  Do
   rv=Elo.GetChildNode              'Read successor node...
   If (rv>0) Then                   '...as long as available
     If Elo.NodeUser=3 Then         'Only user no. 3 receives the document...
       Elo.NodeAllowActivate=1
     Else                          '...none of the others
       Elo.NodeAllowActivate=0
     End If
   End If
  Loop Until rv<0
  Elo.SelectCurrentNode       'Switch back to current node
End If
```

1.1.19 Script event "When creating/moving a reference"

You can use the "When creating/moving an object reference" script event to react when a user moves a document or folder in the repository. For example, you can make changes to the permissions structure or keywording data.

```
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
```

The action key indicates whether an object was recently added to the repository (1), copied (2), or moved (3). Additional parameters, such as ObjectId and parent, can be identified via the OLE interface. This call is only activated when the move/insert action is performed via the client. If the operation is performed via the OLE interface, the script event is not triggered.

This script event is available in versions 3.00.584 and higher and 4.00.182 and higher.

It is also run when folders are copied or added to the repository. In this case, the "ActionKey" property has a value of 4. If the "ScriptActionKey" property is set to -30 in the script, the insert action is canceled.

This script event is available in version 8.00.056 and higher.

1.1.20 Script event "When checking a document in or out"

This script event supplies notifications related to documents to be checked in or out at various times. The individual times are specified by the ActionKey. The following values are available (note that values from 1001 to 1005 are only available in versions 5.00.020 and higher)

Value	Action
30	After checking out a document, but before activating the editor application. The CheckInOutFileName property contains the file name. The ScriptActionKey property indicates whether a document was created from a document template directly in the Repository work area. In this case, bit 8 (value = 256) is set to 1. This property also contains information on whether the document should be opened afterwards with ShellExecute. 0: don't open, 1: open without confirmation, 2: open after confirmation. You can also change this value in the script, which alters the intended view mode.
31	Immediately before checking in a document. The CheckInOutFileName property contains the name of the file to be checked in.
32	After checking in the document. The CheckInOutFileName property contains the name of the file to be checked in, which is deleted immediately after the event.
33	Before checking out a document. The CheckInOutObjID property contains the ELO object ID of the document to be checked out. With the ScriptActionKey property, you can specify whether ELO should continue processing after the script call. If you leave the property unchanged at –1, the process is continued. If you enter a value of 14, ELO assumes that the script has completely processed the check-out operation and that no further actions are necessary. All other values stop the workflow and open a corresponding message box.
34	Before checking in a document. The CheckInOutFileName property contains the name of the file to be checked in.

	With the ScriptActionKey property, you can specify whether ELO should continue processing after the script call. If you leave the property unchanged at –1, the process is continued. If you enter a value of 14, ELO assumes that the script has completely processed the check-out operation and that no further actions are necessary. All other values stop the workflow and open a corresponding message box.
1001	Before displaying the list of files when checking a folder in or out. This event is called for each file in the folder. The file name is contained in the CheckInOutFileName property, while the CheckInOutObjId contains the ELO object ID. The ScriptActionKey property contains information on how the operation is planned (lowest 8 bits), and whether a check in (0x200) or check out (0x100) process is active. In addition, the 0x10000000 bit is set. If this bit is set to 0 by the script event, the file is not displayed in the list. Warning: This script event may be called multiple times in the same operation, such as when a user changes a setting in the dialog box, causing the displayed list to be generated again.
1002	Before checking out a document from the folder list. The additional parameters are set as in event 1001. If the 0x10000000 ScriptActionKey bit is reset to 0, the check-out process is suppressed for that document.
1003	After checking out a document from the folder list.
1004	Before checking in a document from the folder list. The additional parameters are set as in event 1001. If the 0x10000000 ScriptActionKey bit is reset to 0, the check-in process is suppressed for that document.
1005	After checking in a document from the folder list.

Example:

```
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 cannot be checked out"
    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 can no longer be checked in."
    Elo.ScriptActionKey=14
```

```
  end if
end if
```

1.1.21 Script event "When editing the keywording"

This script event contains a number of actions that are distinguished from one another via the ActionKey. When the keywording dialog box opens, it first prompts which index fields should be given a custom button (24). Next, a notification is sent when the keywording is opened (20). A notification is also sent every time an index field is entered or left, as well as when changing the document type. An additional event is triggered when the keywording dialog is closed.

Event 24 must be used if custom action buttons need to be shown at the end of an index field. This event expects a vector with a list of all buttons in the TextParam property. This vector may consist of up to 54 entries with a value of '0' or '1', with each entry corresponding to an index field.

The first character is used for the short name, the second for the full text entry in the search, and the next two are reserved for future functions. The remaining 50 are for the 50 index fields. Therefore, if you require a button by the short name and index fields 2 and 4, you must enter the value 10000101 to TextParam. It is only necessary to specify values up to the final 1, so all zeros at the end can be left off.

When a user clicks one of these buttons, the event will send an ActionKey value of 3xxx - the first index field will be 3000, and the second 3001. The short name returns a value of 3999. Note the additional offset that is entered when the form is opened in the Search work area.

Normally, the buttons are used depending on the keywording form in use. For this reason, the display vector is not only queried when the form is first displayed, but also any time the document type is changed.

Starting with ELO Client Version 6.00.160:

1. When dragging an entry from Outlook to ELO, there is a client setting to specify whether the keywording form should be shown. In order for scripts to recognize whether the dialog box should be shown for the "When editing the keywording" event, a "DropMode" cookie is set when an entry is dragged to ELO. This cookie can have the following content:
    - "Repository Outlook Attachment" : An Outlook attachment was moved to the repository.
    - "Repository Outlook Mail" : An Outlook e-mail was moved to the repository. This entry is also assigned the configured e-mail form, with the Sender and Recipient index fields filled in.
    - "Intray Outlook Attachment": An Outlook attachment was placed in the Intray.
    - "Intray Outlook Mail": An Outlook e-mail was placed in the Intray.
    - "Repository File": A file was dragged and dropped onto the ELO repository.
    - "Intray File": A file was dragged and dropped onto the Intray. When this action occurs, the keywording dialog box is not opened, meaning that the "when editing the keywording" event is not activated either.

Requirement: The option "Show keywording form for Outlook documents dragged into ELO" must be enabled under "Configuration > E-Mail/Signature"

2. Version 7.00.066 and higher: In addition to item 1), the property 'ObjMainParent' is completed during drag-and-drop.

3. An extension to the "When editing the keywording" script event. In this version and higher, additional index fields can be displayed and edited when forwarding a workflow. For this reason, the "Forward workflow" dialog box can also trigger the event.

- If index fields are defined in a person node, a script is called with ActionKey 28 before the dialog is displayed. The keywording data is loaded at this point (short name, index fields, etc.) In addition, the names of the configured index fields can be read and edited in the "TextParam".

  If the user clicks "OK" to forward the workflow and index fields are defined, and the user makes an entry to one or more of the index fields, the script is called at the end with ActionKey 29 before the index field data is saved. The names of the configured index fields are stored in TextParam.

In version 7.00.056 and higher, forwarding can be canceled. To do this, set "ScriptActionKey" to a value of "1". To show the user a message explaining why forwarding is not carried out, the following line of code must be added:

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

Version 7.00.066 and higher

- When the keywording dialog box is closed, the "When editing the keywording" script event is called with ActionKey 25. This always occurs, even if the dialog box is closed without pressing "Cancel" or "OK" (such as by pressing Alt+F4).
- The "When editing the keywording" script event is also called when performing batch keywording.
- Double-clicking the keyword list icon in the keywording now also returns an "Enter" in the OLE interface before the keyword is entered and triggers an "Exit" event after the keyword is input.

From version 7.00.074

- Restricting the list of forms in the keywording dialog
  Two functions are available to provide the user with a shortened list of available keywording forms. One function queries the list of forms, and the other removes forms from the list in the keywording dialog box. These functions are used in ActionKey 24 (defining buttons for display). The following script removes all keywording forms that contain "search" in their name:

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

1.1.22 Script event "Before the search"

The "Before the search" script event allows you to change the SQL search query before it is sent to the SQL server. The SQL command is sent in the "TextParam" property, and changes made by the script are then applied by ELO.

Please note that this script is run every time a search is performed, including for internal search queries (such as when generating lists of links). For this reason, always check whether the desired query is active before changing the search request.

1.1.23 Script event "Viewer export"

With the "Viewer export" script event, you can intervene in the export process in several places. The different times can be identified using the ActionKey property. The target directory for the viewer can found in the ActivePostFile property. You can also cancel the process before completion via the ScriptActionKey property. As long as you leave this value unchanged at 1, the process continues. If the script enters a value of 0, the export is canceled.

"Time" 2 is a good point to copy keyword lists. In ELOprofessional 3.0 you can execute the commands for copying the desired keyword lists here. This can also be manually configured by the user in 4.0. However, at this point you can also use script settings to replace user settings with custom monitoring.

ActionKey	Time
1	Before copying the viewer data. If you interrupt the process here, the result is the same as if the viewer had never been selected.
2	After copying the viewer data and before starting the viewer import. Canceling the process at this point means the general viewer files are copied, but the export data set is not read.
3	After starting the viewer. Please note that at this moment, the actual import process in the viewer is not yet complete. Unfortunately, there is no easy way to determine this time via a script.

4	In ELOprofessional 4.0 and higher: Before copying a keyword list; the file name and path can be found in the ActivePostFile property. If you set the ScriptActionKey to 0, this file is not copied with the others. This lets you control which lists are permitted to be copied (assuming the user has selected the option to copy keyword lists).

Example:

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

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

1.1.24 Script event "Batch scanning"

The "Batch scanning" script event is activated at various times. It is able to influence joined or split documents, as well as the actual filing process. The keywording form for batch filing is determined via the "Set keywording form as default" function in the Intray (similarly to barcode filing). The normal filing destination is also defined here, meaning the keywording form needs to have a filing definition.

This event is available in version 4.00.042 and higher.

1.1.24.1 ActionKey=1: When filing to the repository

This call occurs for every (already joined) document immediately before it is filed. The ActivePostFile property points to the scanned file, MainParentId to the planned filing location, and the other keywording form-related properties (e.g. short name, index fields) are already loaded as well. You can use the script to make any changes in the keywording at this time. In addition, the filing location can be changed.

This call also allows you to implement entirely script-controlled filing. In this case, the script is responsible for the actual filing process, as well as for deleting the document and keywording file. The client signals this event by setting the ScriptActionKey property to a value other than 0.

1.1.24.2 ActionKey=2: During preprocessing

After the scan process, a barcode analysis is performed automatically (if a barcode is defined for the keywording form). Next, a decision is made regarding the first and subsequent pages of the documents. By default, each page with a barcode is a start page, all others are subsequent pages. The script query is performed for all pages. By setting the DocKind property (60: start page, unread, 61: subsequent page, unread, 62: start page, read, 63: subsequent page, read) the script can make its own divisions.

1.1.24.3 ActionKey=3: Before joining

This call is sent before documents are joined. It takes place when the user wants to transfer the documents to the repository via "A" or "S".

1.1.24.4 ActionKey=4: Before filing to the repository

After joining, this call sends an additional notification to the script before the document is transferred to the repository. Actions can be performed at this point that affect the entire batch.

ActionKey=5: After filing to the repository

This call can perform a final action at the end of the process, such as writing a report.

Example:

The following example does not require the barcode components. Instead, an OCR process is used for document analysis (only possible with the full text option). You can easily create test documents for this script by printing pages 100-200 of this document. By detecting the words "Property" or "Function" in the document header, the script detects start pages. All other pages are declared as subsequent pages. Choose a keywording form that does not contain a barcode definition. The filing target, however, must be configured.

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

' nothing happens while filing - the normal filing location
' from the keywording form definition is used
if Elo.ActionKey=1 then
  ' show the currently edited document in the status bar
  Elo.Status Elo.ObjShort
end if

' since the sample documents have no barcode,
' the script must perform all divisions
' into start/subsequent pages
if Elo.ActionKey=2 then
  Elo.Status "OCR processing " & Elo.ActivePostFile
  ' evaluate document header via OCR
  call ELO.OcrClearRect()
  call ELO.OcrAddRect("100,80,999,200")
  call ELO.OcrAnalyze(Elo.ActivePostFile,0)
  'MsgBox Elo.OcrGetText(0)

  ' If the header text contains the text "function", followed
  ' by a function name, then it is a start page
  Cnt=Elo.OcrPattern(10,"*'Funktion'_*L*", Elo.OcrGetText(0))
  if Cnt>0 then
    Elo.ObjShort=Elo.OcrGetPattern(3)
    Elo.DocKind=60
  else
    ' Or if the head contains the text "property" followed
  ' by a function name, then it is also a start page
    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="Subsequent page"
      Elo.ObjMemo=Elo.OcrGetText(0)
      Elo.DocKind=61
    end if
  end if
end if
```

1.2 Script event "HTML keywording display"

ELO version 4.00.080 and higher contain an option to display keywording data parallel to the document. You can customize how it is displayed using an HTML document. To create this document for keywording form number X, create a file named templ_X.htm (such as templ_23.htm for form number 23) in the Intray directory. You can also create a file named templ_default.htm, which applies to all keywording forms that do not require a specific definition. Corresponding wildcards are stored for the current data in these files (in the form of an HTML comment so that the pages can be created with any HTML editor). These wildcards are then replaced with the correct data and shown in a browser window in the ELO client.

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

The two lines shown above show the name (<!--ELO_N_1-->) and the content (<!--ELO_T_1-->) of index field 1 in two cells of a table row. All ELO wildcards begin with an HTML comment introduction <!--, followed by the fixed text ELO_. This is followed by information on whether it concerns the name (N) or the content (T). At the end is the number of the index field and the comment is closed.

The index field identifier can be a number from 1 to 50 (for the 50 index fields). The following letters are available as well:

A	Filing date	<!–ELO_T_A-->
B	Internal ELO number for the attachment file	<!–ELO_T_B-->
D	Document date	<!–ELO_T_D-->
E	Name of the owner	<!–ELO_T_E-->
I	Internal ELO number of the document file	<!–ELO_T_I-->
K	Short name	<!–ELO_T_K-->
M	Keywording form name	<!–ELO_T_M-->
O	Internal ELO number of the logical entry	<!–ELO_T_O-->
S	Internal ELO number of the signature file	<!–ELO_T_S-->
T	Document type	<!–ELO_T_T-->
V	Expiration date	<!–ELO_T_V-->

```
You also have the ability to leave out parts of the HTML document completely,
depending on whether or not a value is entered. Particularly for index fields, a
large amount of space can be saved by not displaying empty values. To do this,
place the complete region within the comment designators <!--ELO_B_xxx--> and <!--
ELO_E_xxx--> (xxx stands for the index field number or one of the special
identifiers previously mentioned).
...
<!--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 this example, index field 1 is included as a table row only if the text in this row is not empty. For the numerical fields with the internal ELO object numbers, 0 (no document assigned) is classified as "empty". Table rows marked as invisible or those to which the user does not have read access are not displayed either.

All permitted HTML constructs (including CSS and JavaScript) are generally allowed in the HTML template file. However, keep in mind that scripting functions are disabled in some browsers. Please also note that the source is a file and does not come from a server, meaning that active contents (active server pages, server-side includes, and so on) will not be processed.

The script event is called before the template file is loaded. At the time of the call, the normal object properties for the document to be displayed are loaded as when following a PrepareObjectEx (Id…). The ViewFileName property contains the name of the template file to be loaded (such as c:\temp\templ_7.htm). In the script, you can now redirect the template file to another file (by setting the ViewFileName property). In addition, you can change the values of the index fields as needed (SetObjAttrib…).

1.2.1 Example: Displays the short name and the first 11 index fields

```
<html><head>
<title>ELOprofessional default form</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 Special script events

ELO recognizes a number of specific scripts that do not need to be configured. Instead, they are recognized based on their names. As soon as such a script is placed in the ELOScripts directory and given a corresponding name, all ELO clients call this directory when the corresponding action is run.

1.3.1 Freezing documents (ELO_Freeze)

The ELO client has a function for freezing documents. This function switches the default printer to the ELO TIFF printer, reads the document, and prints it in Windows using ShellExecute ("print"...). If the ShellExecute command does not process this correctly or completely, you can also automate printing via a script. To do so, create a script named ELO_Freeze that carries out the respective actions.

Before the call, the client sets the file name of the document to be printed as the ActivePostFile parameter. The script uses the ActionKey parameter to signify whether it will handle the printing task itself. If the value is set to 0 (default setting), this means that ELO output the document. Value 1 confirms that the script has initiated printing and value 2 is an error message, which cancels the operation.

Example:

For XLS files, only the current table is printed by Excel via ShellExecute. If the document contains several tables, all tables after the first are ignored. The following script ensures all tables are printed:

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

Elo.Status "File: " & 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 is not restricted to a single thesaurus - it can manage any number of thesauruses. You can use a script to select which thesaurus is used in which index field of a keywording form. The script can perform the selection based on the index field (ScriptActionKey property), based on the current user, based on the current keywording form, or based on the group name of the index field (ActivePostFile property). The number of the thesaurus to be used is returned via the ActionKey property.

Example:

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

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

1.3.3 Keyword lists (ELO_BUZZLIST)

The ELO_BUZZLIST script can be used you want to show different keyword lists after an index field, depending on the user or environment. With the ViewFileName parameter, the client notifies the script which keyword list should be displayed by default. This parameter can then be changed by the script and the view can be directed to a different keyword list.

Starting in version 5.0, the keyword lists are stored in the database. In this case, the script is run with slightly different parameters. The identifier for the database query is passed on by the ScriptActionKey property and has a value of 1000 (in the older format, the index field number was sent here). ViewFileName contains the group name of the index field. With an ActionKey value of 1, the call is coming from the normal keywording dialog box; or with a value of 2, it comes from the keywording forms manager or the main menu. By changing the group name, it is possible to use the script to switch to a different keyword list, since the list is collected based on this entry.

Please note that version 5.0 does not distinguish between selecting and editing the list. If a user has the right to edit the keyword list, the user is also able to add new keyword entries directly in the selection box.

Example from version 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("Select group keyword list:")
    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 Automatic actions at startup (ELO_START)

Although explicit events are available for "When entering the repository" and "When leaving the repository", a script has also been defined (ELO_START.VBS) that starts automatically on these events, without needing configuration in the client. The script can differentiate between the two states with the ActionKey property (1: entry, 2: exit). If both the explicit script event and the automatic script are registered, both scripts are run. When the program is started, the automatic script is run first, followed by the explicit script. When the repository is closed, the reverse occurs.

Version 8 of ELO adds "ActionKey = 3". This key is triggered a little later than ActionKey 1, after the client has completed several internal operations. This ActionKey can, for example, switch to a different work area or add script buttons to the ribbon.

1.3.5 Special handling when filing new documents

When a document is added to the repository for the first time, a script can perform filing. The process becomes active as soon as the user adds a new document to the repository, either from the Intray or by dragging the document to the repository from Windows Explorer. Just before ELO shows the keywording form, a check is performed to see whether a script named "ELO_EXT_<extension>" is available on the computer. Special handling can therefore be associated with the document type (for example, "ELO_EXT_DWG" can be used to handle AutoCAD illustrations). If a corresponding script is available, it is started. The *ViewFileName* ELO property contains the name of the file to be stored, and the *ObjMainparent* property contains the ID of the parent folder in which the user has placed the document. The script is then able to perform the entire document filing process on its own. For example, this method can be used to add documents associated with the original document to the repository. The script must contain the keywording data. The *ScriptActionKey* property informs ELO that the filing process has been taken over by the script, so that no further actions are required by ELO. The property is set to -30 to do this.

1.3.6 Suppress "Document from backup" dialog box (ELO_READDOC)

If a document cannot be loaded in the Repository work area, a dialog box appears which lets the user choose between canceling, retrying, and loading the document from backup. You can use the ELO_READDOC script to suppress this dialog box and prompt the user to make a decision with a custom script. The specific behavior is controlled by the ScriptActionKey property. The following values are available:

0: Show normal dialog

1: No dialog box, continue with "Cancel"

2: No dialog box, continue with "retry"

3: No dialog box, continue with "from backup"

Caution: This query runs in a loop until the document is either loaded or the user has decided to cancel. If you always choose "Retry" or "Backup" in the script and the document cannot be loaded, the program remains stuck at this point.

1.3.7 Configuring the search area (ELO_SEARCHTREE)

When the option "Show tree view" is selected in the Search work area, the script is run after the "Select virtual tree" dialog box is displayed.

ActionKey = 1 - when the default folder has been selected (folder structure)

ActionKey = 2 - when a user-defined structure was selected

In this example, the tree structure is output in a message dialog box:

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

If Elo.ActionKey = 1 Then
' The default structure is displayed
ElseIf Elo.ActionKey = 2 Then
```

```
  iRes = Elo.SortParamlist(1)
  For i=1 To Elo.GetParamCount
    s=s & Elo.GetFromParamList(i) & vbCRLF
  Next
  MsgBox "List of the tree nodes" & vbCrLf & s

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

GetParamCount The value contains the number of rows returned

GetFromParamList A row from the 'Paramlist'

The parameter contains the row number

ObjType of the document

ObjShort of the document

ObjShort of the folders

1.3.8 Static script event when saving keywording (ELO_SAVEINDEX)

When saving keyword data, you may often have to perform a simple action, such as filling in the short name using data from the index fields. This can be handled by using the script timing event "When editing the keywording". However, this script is very complex, as it responds to many different events and has to be configured for users.

For such cases, there is a static script event "ELO_SAVEINDEX" in version 7.0 and higher. If a script file named "ELO_SAVEINDEX.VBS" exists, the script is run when the keywording is saved (i.e. when the OK button is clicked), without having to be registered anywhere in the user's configuration. This script event is called up after the "When editing the keywording" event with the "When saving" ActionKey.

For example, if the entry uses keywording form number 20 and the short name field is empty, it can be filled using a combination of fixed text and partial contents from the index fields:

```
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
```

Information:

The script is run immediately after the script "When editing the keywording - end" is run.

1.3.9 Own reports

You can select various default activity reports from the context menu in the Repository work area. In addition, you can create up to four custom reports here. In this case, it is possible to create any kind of report that is generated using a WHERE clause in an SQL command.

The custom reports are selected via a script. For this purpose, you must create an entry with the name ELO_ACTSELECT. This contains two actions for each of the four possible reports: a) output the name in the menu and b) compose the SQL command. For this purpose, the values 1-4 are transferred for a) and values 101-104 for b) in the "ActionKey" property. The script returns the desired value in the ViewFileName property.

Example:

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

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

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

  case 102
    Elo.ViewFileName="Back today"

  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 Calling scripts

Scripts created within the script manager can be called in various ways:

1.3.10.1 -Run from the ribbon context menu:

Right-click the ribbon in any of ELO's work areas to open a menu where you can select and run your scripts.

1.3.10.2 -Run via user button:

In each work area, eight buttons are available that can each be assigned to a script. First open the script menu (by right-clicking the ribbon) when the cursor is located on one of the tabs. If a menu item (=script name) is selected while holding down the Ctrl key, this script is placed on the ribbon in the current tab. The current assignment is shown when the mouse is positioned above the button. If a script assignment needs to be removed from the button, click the button while holding down the Ctrl key.

After installing ELO, the buttons are not visible by default and may have to be enabled with the *Configure toolbar view* function.

In order to display an icon on the script button, a 24 x 24 pixel BMP file must be created and stored with the same name as the script in the ELO scripts directory.

1.3.10.3 -Run via context menu:

Script calls can be added to the context menu of any work area. If you want to add a script to the context menu of a specific work area, first select the work area. Next, add a script to the context menu by right-clicking the ribbon and clicking the script while holding down the *Shift* and *Ctrl* keys. To remove a script from the context menu, select the script in the context menu while holding down the *Ctrl* key.

1.3.10.4 -Run via ELO menu items or buttons:

You can add a custom script to every menu item and button in the ELO work areas, which is then called instead of the original function. You can specify within the script whether the original ELO function should remain activated after the script is finished or if it should be ignored.

You can assign scripts via the internal name of a menu item or button. The name of the script must start with an "@" and be followed by the name of the control (menu item or button). A list of the controls can be found in the appendix of the ELO Automation Interface documentation.

When the script is started for the first time, the *ActionKey* property has a value of 40. The script can control downstream processes depending on the configuration of the *ScriptActionKey* property:

20 = the original ELO function will be activated at the end of the script

21 = the original ELO function will be activated at the end of a script and then the script will be started again with *ActionKey 0*.

1.4 Keywording form recognition API

1.4.1 Overview

Form recognition in ELO requires two to four steps. In the first step, the rectangle areas of the document are selected, which are necessary to classify the document type. Typical areas here are the texts "invoice" or "delivery note" or a company name (no graphical elements). These areas are then processed, recognized, and saved to a text list by the OCR software. The extended OCR API was developed for this purpose in version 2.05.104.

In the second step, a search is performed for characteristic patterns for every document type in the recognized text. If a certain invoice has the text "invoice" followed by an invoice number at a defined place (in a rectangle defined for the upper right, for example), the invoice text only has to be checked for this pattern in order to detect the correct keywording form. For this, the pattern recognition API (also new from version 2.05.104) is used.

Once the keywording form type is recognized, it may prove necessary to detect additional text areas. This means that in addition to determining if a document is an invoice, you may want to detect the order number, customer number, or invoice amount. This step can, in theory, occur right at the start. However, if you do this, you must also check all defined rectangle areas to exclude the possibility of other document types in the same step. As this can be very time-intensive, it usually makes sense to move this back to a separate third step if you have a large number of document types.

After all text regions are read and the document type has been determined, the keywording data is extracted from the text blocks. During this fourth step (which in simple applications may have already been covered in step two), the pattern recognition API is used again.

1.4.2 Pattern recognition API commands

Function: OcrAddRect

Function: OcrAnalyze

Function: OcrClearRect

Function: OcrGetPattern

Function: OcrGetText

Function: OcrPattern

The use of these commands is demonstrated in the following examples. Please refer to the command list for the exact parameters.

1.4.3 Examples

1.4.3.1 Example 1: Invoice recognition

This first example is based on a simple case. There are only two possible document types, an invoice and everything else. Only the invoice number needs to be entered to the keywording.

In the first step, the rectangle list with the regions for the invoice test and corresponding number is set by the OCR API and the recognition process starts.

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

In the second step, a check is performed to detect whether the form is an invoice. This step checks whether in text block 1, the text "invoice" can be detected followed by an invoice number.

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

The step searches for a pattern consisting of five parts:

An arbitrary string of prefix characters, which can be empty (the rectangle may include foreign characters due to the scanner feed not working completely cleanly).

The text "invoice"

A string of empty spaces of arbitrary length, including spaces

A number (the invoice number)

Any additional text (for example, text that does not belong to the expression but that is in the detected rectangle anyway).

If the pattern was recognized, the function returns the value 5 (=number of pattern parts). If an error occurs, a negative value is returned. After this, the text for the individual pattern parts is available in a text field.

```
If CntRechnung=5 then
   ’ it is an invoice, 5 pattern blocks were detected
    ELO.PrepareObjectEx(0,254,RechnungsMaskNo)
    ELO.SetObjAttrib(0, ELO.OcrGetPattern(3))
      …
end if
```

The four-step process is reduced to two steps in this simple example. Step 3 does not apply since no further OCR areas have to be read and step 4 does not apply as the keywording was already detected during classification. By checking whether the document was an invoice, the invoice number was immediately read into the sample text field and can be used right away.

1.4.3.2 Example 2: Invoice/Delivery note detection

This example is based on a simple case. There are only two possible document types: an invoice and a delivery note. Only the invoice number or the delivery note number need to be entered to the keywording, respectively.

In the first step, the rectangle list with the regions for the invoice/delivery note text with the appropriate number is set by the OCR API and the recognition process starts.

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

```
x=Elo.OcrAnalyze(FileName,0)
```

Then, a check is performed whether the form is an invoice or a delivery note. This check verifies whether the text "invoice" can be detected in text block 1 or if the text "delivery note" is in text block 2.

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

If CntRechnung<0 then
   ’ it is not an invoice
  If CntLieferschein<0 then
    ’ it is not a delivery note either, so nothing is done
    DocType=0
  else
    ’ delivery note
    DocType=1
  End if
Else
  ’ it is an invoice
  if CntLieferschein<0 then
    ’ it is really only an invoice
    DocType=2
  Else
    ’ it is an invoice and a delivery note - there is an error here
    DocType=0
  End if
End if
```

Now, the keywording is entered. Additional rectangles are not included in this simple example since the invoice or delivery note number is already detected by OCR.

```
Select Case DocType
  case 1 ‘Lieferschein
    ELO.PrepareObjectEx(0,254,LieferscheinMaskNo)
    CntDeliveryNote=Analyze( “*’Delivery note’_N*“, OcrText(1) )
    If CntLieferschein=5 then
      ELO.SetObjAttrib(0, Elo.OcrGetPattern (3))
      ...
    end if
  case 2 ‘Invoice
    ELO.PrepareObjectEx(0,254,InvoiceMaskNo)
    CntInvoice=Analyze(  “*’Invoice’_N*“, OcrText(0) )
    If CntRechnung=5 then
      ELO.SetObjAttrib(0, Elo.OcrGetPattern (3))
      …
    end if
end select
```

1.4.3.3 Example 3: Form recognition trade fair demo

The following example is a complete script for detecting three different forms and automatic filing in the repository (the folder is created if necessary).

```
'OCRELO.VBS  24.08.2000
'-----------------------------------------------------------------
' © 2000 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo.info)
'-----------------------------------------------------------------
' This script analyzes the Intray documents for certain texts
' that indicate invoice numbers and then stores the
' recognized invoices in the corresponding folders
' (which are automatically created, if needed)
'
' -----------------------------------------------------------------

set Elo=CreateObject("ELO.professional")
MaskNo=Elo.LookupMaskName("ELOInvoice")

' run through all Intray entries (at most 200)
Elo.SelectView(3)

' first, analyze all documents
Elo.Status "Analyze documents"
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

' and now transfer the detected documents to the repository
for j=i to 0 step -1
  Move(j)
next

' take some time to clean up the Intray ...
x=Elo.UpdatePostboxEx(0,0)
Elo.Status("Finished")
' ... finished
```

```
' Help functions
'
' this function transfers a detected document to the repository
sub Move( iPostLine )
  res=Elo.PrepareObject( -1, iPostLine, 0 )
  if res>0 then
    Text=Elo.ObjShort
    Elo.Status("File: " & 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
        Ivno=Ivno & " (Credit note)"
      else
        Ivno=Ivno & " (Invoice)"
      end if
      Elo.ObjShort=Renr
      iRet=Elo.AddPostBoxFile("")
      RegId=CheckFolder( Date, CuNo )
      if RegId>0 then
        x=Elo.MoveToArchive( "#" & RegId )
      end if
    end if
  end if
end sub

' this function checks whether the target folder exists for a document
' and creates it, if required
function CheckFolder( Date, Customerno )
  RegId=Elo.LookupIndex( "RELO=" & Right(Date,4) & ":" & Customerno )
  if RegId<1 then
    ' if folder does not exist, it is now created
    FolderId=Elo.LookupIndex( "¶ELO Invoices¶" & Right(Date,4) )
    if OrdnerId>0 then
      ' folder found; now creating the child folder
      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

' this function executes an OCR analysis for the current
' Intray document
```

```
' and adds the invoice number to the short name if a
' known document type was found
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,"'Invoice'L'Number:'NL'Date:'*L'Orderno.:'NL'Customer-
No.:'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 credit note
    x=Elo.OcrPattern( 10,"*'Credit note'L'Number:'NL'Date:'*L'Order-
No.:'NL'Customer-No.:'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 invoice
    x=Elo.OcrPattern( 10,"*'Invoice'n*'Customernumber'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 "Detect  line " & i & " : " & Elo.ObjShort
end sub
```

1.4.4 Syntax of the pattern recognition format strings

Only one format string must be provided for pattern recognition. In a single step, ELO can check whether the text corresponds to the pattern, then divide the text into the pattern parts. Please note that a pattern may be made up of a maximum of 32 parts (in version 104, this may possibly change later). The following pattern parts are available:

*	Any text	This partial pattern accepts any text, but it can also be empty. In addition, you can specify a length before the asterisk. The recognized text has to be at least this long.
_	Space	This partial pattern accepts any character string (even if empty). Specifying a length before the underscore will force a string to be recognized with the exact indicated length.
L	Line break	This partial pattern detects exactly one line break. A number in front of the L (e.g. 3L) requires exactly this number of line breaks.
N	Number	An arbitrary string of numbers is recognized (but no empty string). The first character that is not a number ends this string (line breaks or space characters as well). If a number is placed before the N, a character string of exactly this length is recognized. Example: ABC12345XYZ *N* detects [ABC][12345][XYZ] *3N* detects [ABC][123][45XYZ] *6N* detects nothing, because there is no numeric string of this length.
n	Number (special OCR)	Similar to N (Number) – the difference is that this format also accepts some letters that are similar to certain digits (O, o, and Q are recognized as a 0 (zero), I and l are detected as 1 (one).

"..."	Text	The text between the quote marks is accepted. However, the text must exist in exactly this format. No additional spaces or line breaks are accepted. Example: xyzInvoice 123 *"Invoice"_N* recognizes [xyz][Invoice][ ][123][] *"Invoice"N* recognizes nothing, because the space character was not considered in the format string *"INVOICE"_N* recognizes nothing, because invoice is written another way.
'...'	Text	Similar to the quote marks, but text recognition is not case sensitive. Using it in the example above, for example, would now recognize the third case as well.

1.4.5 Notes

Please note that the recognition algorithm searches until it reaches a "match" or until it is certain that no match can be found. It does not get stuck on partial hits. A naive implementation based on the pattern "*'invoice'_N*" and the text "xxx invoice copy yyy invoice 12345 zzz" could locate the text "invoice" of the invoice copy and if no number follows, it could give up the search. ELO, however, continues searching after this failed attempt and recognizes the text as expected.

The pattern * in particular requires a high amount of internal resources, since in edge cases many possibilities need to be analyzed. Nevertheless, this operator must often be used. However, take care to frame these expressions in general with *....*, which eliminates "dirty characters" at the beginning or end of the text. These characters are often caused by outside characters that bleed into the detection rectangle. Nevertheless, they should be used only where it is justified. The unnecessary, but seemingly harmless combination ** does not change the recognition results, but it has a significant negative impact on performance. The pattern

```
** zwingt ELO bei dem Text „abcd“ zur Kontrolle der Möglichkeiten [][abcd], [a][bcd], [ab][cd], [abc][d], [abcd][].
```

Do not put too much into one pattern. If you have a rectangle in an invoice that contains successive lines consisting of the invoice number, customer number, order number, and job number, you can search for each of them individually. You could also create a complex search pattern of:

'invoice'_N'customer'_N*'order no.'

In the second case, all numbers would be recognized in one run. But as soon as one of these characters is not recognized correctly by the OCR software (e.g. custorner instead of customer) no more characters will be detected. If you are only interested in keywording the document when everything has been recognized, the second method is appropriate. However, if you are of the view that as much as possible should be detected, and everything else can be manually added later, you should detect the numbers individually.

For complex patterns in particular, the pattern may not be recognized, even though the text actually conforms to the schema. Since no "pattern debugger" exists, it is only possible to use trial and error to improve the results. Beginning with the "defective" pattern:

'invoice'_N'customer'_N*'order no.'N*'Auftrag'_N*

you can check *'invoice'* in

one step. After that, you check

’invoice’_N

’invoice’_N’customer’*

’invoice’_N’customer’_N*

etc. The pattern is always extended by one (or more) step(s). Make sure you always end the pattern with an asterisk (*). This * is the match for the entire rest of the text. If it is missing, nothing will be recognized.

1.5 General information about the functions

The return values of the functions normally contain an error code. In these cases, negative values indicate a function has failed, whereas positive values indicate a function has run correctly.

Internally, ELO uses "object IDs". These are unique values (within a repository), which are allocated to each entry upon creation and via which the entries can be called up again at any time. If you require any references directly from ELO in your program and want to store them, you should use the object ID instead of the name. The object ID remains unchanged for the entire lifetime of the entry. However, the name can be changed by the user or by other programs at any time.

1.6 ActionKey (int, read only) property

Some ELO events combine to trigger a script. In this case, the ActionKey can be used to determine what kind of event has occurred.

Dialog box: Edit document	Dialog box opened in Intray work area, document type not yet defined	19
	Open keywording form	20
	Close form with "OK"	21
Event:	Close form with "Cancel"	22
	Document type changed	23
	Request button list	24
	When the form is closed (last triggered event)	25
After editing a keywording form field	When entering the short name field	10
	When leaving the short name field	11
	When entering the "Extra text" tab	12
	When leaving the "Extra text" field	13
	When entering the "Date" field	14
	When leaving the "Date" field	15
	When entering index field "n"	1000+n (n=0-49)
	When leaving index field "n"	2000+n
	Custom button clicked	3000+n
Before importing/exporting	Before importing a file folder	1
	Before exporting a file folder	2

When checking in/out	After checking out	30
	Before checking in, after requesting version information	31
	After checking in	32
	Before checking out	33
	Before checking in, before requesting version information	34
	Before discarding	38
	Before activating/displaying	80
	Before printing	81
	Check-in/Check-out folder	1001 … 1005
Reading/saving user	Immediately before saving	20000
	After saving	20001
	After reading	20010
Keyword list	Before editing the keyword list	1
	Before displaying the keyword list	2
Reminder	Reminder as task for other user	1
	Reminder as task for oneself	2
	Reminder as e-mail	3
ClickOn event	Selection of a button or menu item	40
Workflow event	Only display workflow deadline in ELO	3
	Workflow deadline as task	4
	Workflow deadline as e-mail	5

Freeze document	Before printing	0
Searching ("Before collecting the search results")	Before the search	1
	F7 group search	2
	Combined group search	3
	After the search	4
When editing the keywording – display of index fields in the "Pass workflow forward" dialog of a workflow	WF: Before showing the pass forward dialog box	28
	WF: After displaying the pass forward dialog box	29

These numbers are overwritten by the value 0x8000 if the keywording form is a search form and not a form used to store data. This information must always be evaluated, as otherwise there may be undesirable effects when trying to search for documents.

An example (stored as AutoShortName script and entered for the "When editing the keywording" script event):

```
' enter the number of the keywording form here
' the script automatically fills in the short name
' using the contents of index fields one and two
DokumentenMaske = 2

Set Elo=CreateObject("ELO.professional")

if Elo.ActionKey=21 and Elo.ObjMaskNo=DokumentenMaske and Elo.ObjShort="" then
  ' writing takes place after closing the keywording dialog box, if the
  ' correct form is selected and no short name has yet been entered.
  Elo.ObjShort=Elo.GetObjAttrib(0) & " " & Elo.GetObjAttrib(1)
end if
```

1.7 ActivePostFile (AnsiString) property

This property specifies the access path and name of the active Intray file. This entry is set through the actions that create a new Intray entry (i.e. by scanning or with the AddPostboxFile function).

In version 3.00.508 and higher, this property can also be written to. This lets you set a file as active if it is already in the Intray.

See also:

- AddPostboxFile
- PrepareObject

1.8 ActiveUserId (int, read only) property

The ActiveUserId property returns the internal ELO user number of the active user.

See also:

- LoadUserName
- FindUser
- SelectUser

1.9 Activity (AnsiString) property

The Activity property sets or reads the current value of an activity. This value is only valid for the event routine "When reading or writing an activity". If the value is requested at other times, there may be errors up to and including a protection violation.

The value of the activity is a string that includes all fields. These are separated by the delimiter character (i.e. ¶).

Available since: 3.00.510

Example:

```
' When writing an activity, "Test99" is entered
' in the comment field;
' when reading, the text version of the activity
' is displayed in a message box.
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 AddAutoDlgControl (int, int, AnsiString, AnsiString) function

Available from: Version 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	Created object	Caption	Default
1	Label	Label text	No function
2	CheckBox	Text behind check box	1 – Checked, otherwise unchecked
3	Radio button	Text behind selection button	1 – Checked, otherwise unchecked
4	Edit field	Text in front of the input field	Text of the edit field

Horizontal position of the object (permitted values: 0 to ?)

Return value:

- 1 – Object was created.

Example:

Creates a dialog box with a label at the top position and an edit field below with entered

content and displays it.

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

See also:

- CreateAutoDlg
- ShowAutoDlg
- GetAutoDlgValue

1.11 AddLink function

Connects two ELO objects (documents or folders) through a logical link. Any entries can be linked, without taking the repository hierarchy into account. One object can also be linked to multiple other objects.

```
int AddLink( int StartObjekt, int Zielobjekt )
```

Parameter:

StartObject: Object ID from which the link is extended

TargetObject: Object ID of the entry to which the link points

Return values:

-2: Error creating the link

-1: No active work area

1: Link added

Available from: 3.00.270

1.12 AddNote function

1.13 AddNoteEx function

1.14 AddNoteEx2 function

1.15 AddPostboxFile function

This function transfers a file to the Intray. First, prepare a new entry with PrepareObject, which sets various properties like the entry's short name. Now, call the function. The file is then **copied** and stored in the Intray together with the keywording.

After running this function, the new entry then automatically becomes the "active" Intray entry, which is then the source for a number of subsequent functions.

There is another special case for this function: if the "active" Intray entry is changed and then needs to be saved again in the Intray, this function is called with an empty string as parameter. This function is often used in barcode processing, for example.

```
int AddPostboxFile( AnsiString SourceFile )
```

Parameter:

- SourceFile file to be copied

Return values:

- -3: Error when saving.
- -2: No active Intray entry available
- -1: Error when transferring to Intray
- 1: Ok

Example:

Call the first Intray document, set the keywording form ID to 2 and automatically fill the short name and first index field. After that, the document is transferred to the repository using the path provided:

```
' Destination folder for the document
FolderId = "¶Folder1¶Folder2¶Folder3"
MaskNo = 2
Set Elo=CreateObject("ELO.professional")

' First, update the Intray to be safe
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 "No document found in the Intray"
end if
```

See also:

- MoveToArchive

- LookupIndex
- UpdateDocument
- InsertAttachment

1.16 AddSignature function

This function attaches an external signature file to an existing document. The script is responsible for ensuring that the signature actually belongs to the document. Otherwise, if errors were to occur, an invalid signature would be displayed to the user.

```
int AddSignature( int ObjectId, AnsiString SignatureFile )
```

Parameter:

- ObjectID Logical ELO document ID
- SignatureFile File with the signature information

Return values:

- 1: Ok
- -1: No open workspace
- -2: Error saving the signature file

Example

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

Available from: 4.00.180

1.17 AddSw function

This function adds a keyword to the keyword list. Every entry of a keyword list is given a unique indicator within its level – a 2-digit letter combination: AA, AB, AC up to ZZ. When entering a new keyword, the next available entry is determined and returned as the result of the call. You will need this value, for example, when you want to create treelike child entries.

The group designates the assignment of a list to an index field (i.e. the name of the keyword list must be the same as the group field in the keywording forms editor). The parent entry provides the access path to the parent node of the new keyword. The root for this is indicated with a period. A parent entry "." therefore creates a keyword at the lowest level, "AA" creates one below the first entry and "AB" creates one below the second entry.

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

Parameter:

- Group Selects a keyword list
- Parent Predecessor node for the new entry
- Word New keyword

Return values:

- -1: No open workspace
- -2: Error saving the keyword
- otherwise: position of the new keyword

Example:

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

Available from: 5.00.066

1.18 AddThesaurus function

This function creates a thesaurus entry in the database. Every thesaurus group has to have a unique group number (group ID). If you transfer a 0 for the first entry of the group, ELO will define a new random number. The "Prio" parameter defines the sorting order, but the ListId parameter currently has to be set permanently to 1 (thesaurus in the keywording dialog box).

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

Parameter:

- ListId Area, set to 1 for the keywording
- GroupId Thesaurus group, 0: acquire a new group
- Prio Sorting order
- Value Text

Return values:

- >0: ok, group ID
- -1: No open workspace
- -2: Error saving the data

Example:

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

Group = Elo.AddThesaurus( 1, 0, 10, "Tool" )
MsgBox Group
if Group > 0 then
  MsgBox Elo.AddThesaurus( 1, Group, 20, "Hammer" )
  MsgBox Elo.AddThesaurus( 1, Group, 30, "Screwdriver" )
  MsgBox Elo.AddThesaurus( 1, Group, 40, "Pliers" )
end if
```

Available from: 4.00.214

1.19 AnalyzeFile (invalid) function

This function sends the content of a file in the Intray to the OCR module and enters the detected sections into predefined keywording form fields. The name of the Intray file can be transferred with SourceFile. Alternatively, a line number (#0, #1, #2, etc.) can be returned. The corresponding file in the Intray list is then used as the file name. After recognition, the result is saved in the keyword file for the image file.

The OcrDescriptor parameter contains the list of the rectangles to be checked and the form fields to be filled. Refer to the barcode documentation for a detailed description of this list.

The rectangle must be defined somewhat differently from the barcode module as follows:

"R(left,top,right,bottom)"

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

Parameter:

- SourceFile The file to be analyzed or the line number in the Intray (with #number).
- OcrDescriptor Rectangle list, in barcode format.
- MaskNo Keywording form number of the keyword file to be created

Return values:

- 1: Ok
- -1: No open workspace
- -2: OCR subsystem not loaded
- -3: No rectangle list available
- -4: Faulty rectangle list
- -5: Error during OCR processing
- -6: Error writing the detected data

See also:

- AddPostboxFile
- ReadBarcodes

1.20 AchiveDepth (int) (invalid) property

With this property, you can determine the number of hierarchy levels of the current repository. The level of the documents is included in the count, meaning a classic ELO repository would have four levels of hierarchy, consisting of the structural elements "folder" -> "folder" -> "folder" -> "document".

If the Repository work area is not active yet, the ArchiveDepth property has the value –1.

1.21 ArcListLineId function

With this function, the ELO object ID can be determined from a line in the repository list in the right-hand pane.

```
int ArcListLineId(int LineNo)
```

Parameter:

- LineNo Line to be selected

Return values:

- -2: Invalid row number
- -1: No active work area
- >0: Object ID

Available since: 5.00.036

Example:

```
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
```

See also:

- UnselectArcListLine
- SelectArcListLine
- ArcListLineSelected

1.22 ArcListLineSelected function

This function is used to check whether a line is selected in the repository list on the right-hand side.

```
int ArcListLineSelected(int LineNo)
```

Parameter:

- LineNo Line to be selected

Return values:

- -2: Invalid row number
- -1: No active work area
- 0: Line not selected
- 1: Line selected

Available since: 5.00.036

Example:

```
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
```

See also:

- UnselectArcListLine
- SelectArcListLine

1.23 AttId (int) property

The AttId property defines the working version of an attachment to a document. This entry must be taken from the list of available attachments. A foreign entry may cause serious malfunctions here.

Available since: 5.00.042

Example:

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

See also:

- MaskKey
- DocKey
- DocKind
- DocPath

1.24 AutoDlgResult (AnsiString) property

This property transfers the result for all objects of the AutoDialog. The individual values are separated by a return.

Example:

- Transfers the content of the edit field.
- Elo.CreateAutoDlg ("New dialog box")
- Elo.AddAutoDlgControl (4,1,"Name","")
- Elo.ShowAutoDialog
- MsgBox Elo.AutoDlgResult

Available from: Version 3.00.228

See also:

- CreateAutoDlg
- AddAutoDlgControl
- ShowAutoDlg
- GetAutoDlgValue

1.25 BringToFront function

The BringToFront function allows you to bring ELO to the foreground on the desktop, making it visible to the user after ELO was previously hidden by other remote-controlled applications.

```
void BringToFront()
```

Parameter:

- None

Return values:

- None

See also:

- EloWindow

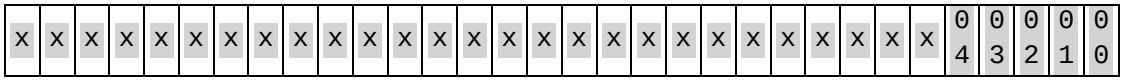
1.26 ChangeObjAcl function

Available from: 3.00.???

This function allows you to assign and/or change access permissions to objects.

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

- ObjId : Internal ELO ID
- Acl : Rights to be allocated. Multiple rights must be separated by a comma.
  - Format : XYZ
  - XY:
    - RW – Read/Write permissions
    - RD – Read permission
    - WT – Write permission
    - KY – Key
  - Z:
    - User number / Group number/ Key number
- Option: 32-bit integer form



- 00 0 – Add rights
- 1 – Overwrite rights
- 01 0 – Inherit to non-child elements
- 1 – Inherit to child elements (not implemented)
- 02 0 – Inherit identical rights (not implemented)
- 1 – Only inherit changes (not implemented)
- 03 0 – Warning dialog box if one's own permissions will be removed
- 1 – Warning dialog box off if one's own permissions will be removed
- 04 0 –
- 1 – Accept parent permissions (not implemented)
- x Currently unused (reserved for extensions)

Return value:

- -1 - No active work area
- -2 - Insufficient rights to carry out changes
- -3 - User cancel in case of one's own rights being removed (dialog box)

Otherwise new element permissions

Example:

Display the current permissions to object with ID 45.

```
Rechte = Elo.ChangeObjAcl (45,"",0)
```

Sets the system key for object 30 and removes all others.

```
Elo.ChangeObjAcl (30,“KY0“,1)
```

1.27 CheckFile function

You can request various file properties with the CheckFile function.

```
Int CheckFile( int OptionNo, AnsiString Dateiname )
```

Parameter:

- OptionNo: 0: Check exclusive access to file
- File name: Name of file to be checked

Return values:

- 1: Ok
- -1: Invalid number under OptionNo
- -2: Exclusive access not possible

Available since: 3.00.288

1.28 CheckFileHash function

With the CheckFileHash function, you can check whether a file has already been filed to ELO. A dialog box is shown to the user, in which the user can decide whether the user wants to file the document anyway, create a reference instead of creating a duplicate, or completely cancel filing.

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

Parameter:

- HeaderMessage: Additional comment in the dialog box title (e.g. file name or description)
- FileName: Name and path of the file to be checked
- ModeFlag: Reserved, must be set to 128

Return values:

- 0: Document is not already present in the repository
- 1: Document exists, file anyway
- >1: Document exists, create reference to this DocID
- -1: No work area active, no check possible
- -2: Document exists, do not file again

Example:

```
Set Elo=CreateObject("ELO.professional")
MsgBox Elo.CheckFileHash( "Testfile", "d:\temp\Scanfile.tif", 128 )
```

Available since: 4.00.034

See also:

- GetMD5Hash
- LookupHistMD5

1.29 CheckIn function

1.30 CheckInEx function

You can use this function to check in a document that you have already downloaded from the repository via CheckOut. Please note that you must not change the file name, as this contains the identifiers for the repository in use and the object number. In ELOprofessional 3.0, the CheckIn command prompts a query for the version number and a comment. You can suppress this query by using the CheckInEx command, which sends this information as parameters instead.

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

Parameter:

- FileName : Name of file to be checked in
- sComment : Comment on the new document version
- sVersion : Version number

Return values:

- 0: Object ID of the document checked in
- -100: No active work area
- -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

Available (Ex) 3.00.278

Example:

The document can be encrypted during check-in.

Note: The old versions of the document remain unencrypted and can be viewed without a password.

```
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,"Automatic encryption","1.0")<0 then
          CryptId=-3 ' Error during CheckIn
        end if
      else
        CryptId=-2 ' Error during CheckOut
      end if
```

```
    else
      CryptIt=2 ' is already encrypted, nothing to do...
    end if
  end if
end function
```

1.31 CheckInOutFileName (AnsiString) property

With this property, you can determine the file name of the document within the "When checking a document in/out" event. If required, the file name can be changed for the "Before checking in" event.

See also:

- CheckInOutObjID
- ActionKey

1.32 CheckInOutObjID (int) property

With this property, you can determine the object ID of the document within the "When checking a document in/out" event.

See also:

- CheckInOutFileName
- ActionKey

1.33 CheckObjAcl function

With the function CheckObjAcl, you can check which permissions you have to a specific object. If the object is already active (for example, within a keywording event or after a PrepareObjectEx), you can enter 0 for the lObjectId. In this case, the current value from the ObjAcl property is used. If the object is not active yet, you can also specify the ObjectId as a parameter. Then, the ACL list of the object is loaded and checked. As only one property is read from the database, and not the entire object, this solution is faster than combining PrepareObjectEx (Id…) and CheckObjAcl(0). This only applies if the script does not require a PrepareObjectEx anywhere else.

```
AnsiString CheckObjAcl( int lObjectId )
```

Parameter:

- lObjectId Number of the document to be checked (internal ELO ID) or 0

Return

- -1 No active work area
- -2 Error reading the ACL
- >=0 Authorization form (1: Read, 2: Write, 4: Delete, 8: Edit Document)

Available since: 4.00.178

Example:

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

' Only ObjectId known:
MsgBox Elo.CheckObjAcl( Id )


if Elo.PrepareObjectEx( Id, 0, 0 ) then
  ' The object is already loaded:
  MsgBox Elo.CheckObjAcl( 0 )
end if
```

1.34 CheckOut function

With the check-out function, you can remove a document from the current repository for editing. After you are done editing, you can return the document back to the repository using the check-in function. Please note that you must not change the file name, as this contains the identifiers for the repository in use and the object number.

```
AnsiString CheckOut( int lObjectId, int iMode )
```

Parameter:

- lObjectId Number of the document to be checked out (internal ELO ID)
- iMode 0 Check out only, do not activate application
- 1 Check out and activate application without request
- 2 Check out and activate application after confirmation

Return value:

- File name of the checked out document

1.35 CheckPage function

With the CheckPage function, you can check whether a separator or empty page exists for the current Intray entry. You can find an example for this command in the prefix.

Int CheckOut (int Mode, int hit ratio)

Parameter:

- Mode 1: Empty page control 2: Separator page control (combinations also permitted)
- Hit ratio 0...999 required hits per thousand. A value of 970 (default in the ELO client) requires a hit ratio of 97%.

Available from: 3.00.220

1.36 CheckUpdate function

With this function, you can decide whether a change is displayed immediately. For automatic capture of mass data especially, ELO not showing each change can save a great deal of time. In this case, use an ELO.CheckUpdate(0) before the run and an ELO.CheckUpdate(1) after the run.

```
int CheckUpdate( int DoCheck )
```

Parameter:

- DoCheck : 0: No display, 1: Display

Return values:

- Old status

See also:

- EloWindow

1.37 ClickOn function

With this function, you can simulate clicking a dialog element in the main ELO dialog box. This allows you to call all functions that are available in the menus or toolbar.

If using the "ClickOn" function with a function from the context menu, the object ID of the ELO entry to be edited must be assigned to the PopupObjID property.

```
int ClickOn( AnsiString ComponentName )
```

Parameter:

- ComponentName: Name of the dialog component (list of all dialog items see appendix A)

Return values:

- -1 Main dialog box not active
- -2 Component not available
- -3 Component does not have OnClick routine
- -4 Component type is not supported
- -5 Function can currently not be requested (dialog component "disabled")
- 1 OK

1.38 CloseActivateDocDlg (invalid) function

With the CloseActivateDocDlg function, you can check in a document you checked out to edit back into ELO. Choose from the options mode=1: Ok, accept new version and mode=2: Cancel: Discard changes

```
int CloseActivateDocDlg( int Mode )
```

Parameter:

- Mode 1: Ok, 2: Cancel

Return values:

- -1 ActivateDocDlg not active
- -2 False mode parameter
- -3 Error closing the dialog box
- 1 OK

1.39 CollectChildList function

With the CollectChildList function, you can determine all subsequent nodes for an object ID. The node list is transferred as text string, the individual successor IDs are separated by colons.

```
AnsiString CollectChildList( int ObjId )
```

Parameter:

- ObjId Node number of the start object

Return values:

- empty:Error, not specified in more detail
- else: successor list

1.40 CollectLinks function

With the CollectLinks function, you can determine all links for an object ID. The IDs of the linked objects are returned as a comma list.

```
AnsiString CollectLinks(int iObjID, int iMode)
```

Parameter:

- iObjId Object ID of the ELO object
- iMode 0

Return values:

- "-1" No work area active
- empty: no links available
- else: successor list (object IDs separated by comma)

Available since: 3.00.378

1.41 ColorInfo (int) property

The ColorInfo property determines the color value of a color definition. This color value is transferred in standard Windows RGB style.

See also:

- ReadColorInfo
- WriteColorInfo
- ColorName

1.42 ColorName (AnsiString) property

The ColorName property determines the color name (short name) of a color definition.

Maximum length: 30 characters

See also:

- ReadColorInfo
- WriteColorInfo
- ColorInfo

1.43 ColorNo (int) property

The ColorNo property determines the color number of a color definition.

See also:

- ReadColorInfo
- WriteColorInfo
- ColorName

1.44 CreateAutoDlg (AnsiString Caption) function

Creates an empty dialog box with an Ok & Cancel button.

```
int CreateAutoDlg (AnsiString Caption)
```

Caption:

- Title of the dialog box

Return value:

- 1 – dialog box was created.

Available since: 3.00.228

Example:

```
An empty dialog box is displayed.
Elo.CreateAutoDlg ("An empty dialog box")
Elo.ShowAutoDlg
```

See also:

- AddAutoDlgControl
- ShowAutoDlg
- GetAutoDlgValue

1.45 CreateCounter function

The CreateCounter function creates a new counter with a start value that can be preset.

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

Parameter:

- CounterName Name of counter to be created
- InitialValue Start value

Return values:

- -1: Error
- 1: Ok.

See also:

- GetCounterList
- GetCounter

1.46 CreateStructure (AnsiString Path, int StartID) function

Available from: Version 3.00.228

This function creates a list of folders from the current position, or if a separator is set at the start, starting from the top-level folder. Existing paths with the same name are only reinitialized if there is a separator at the end of the path.

Version 8.00.056 and higher

A "closing" pilcrow (¶) will now be ignored. Some Business Partners encountered problems when the "index string" consisted of variables and one of the variables did not contain a value.

```
(AnsiString) CreateStructure (AnsiString Path, int StartID)
```

Path: Path to be created

Individual elements are to be separated by the separator

Separator at start Create from top-level folder position

StartID is ignored

Separator at the end Existing paths are overwritten

StartID: Valid values: ObjectID, 0, negative ObjectID

0 – Creation of path within currently selected object

A negative StartID results in the creation of a positive StartID

at the same level.

The separator at the start of a path has precedence over the StartID. (StartID is ignored)

Return values:

List of ObjectIDs of created elements, separated by the separator (235¶252¶22)

- -6 - "Edit repository" right missing (starting version 6.00.078)
- -7 - The user does not have the "Edit lists" right to one of the existing folders of the path provided (starting with version 6.00.078)
- -5 – No entry selected
- -4 – No valid start position given
- -3 – Empty field name
- -2 – No active work area
- -1 – Too many objects (repository depth is exceeded)

Example:

Creates a top-level folder, a second-level folder, and a third-level folder. The starting position is the top folder level.

ELO. CreateStructure ("¶My_Folder1¶My_Folder2¶My_Folder3",0)

Creates a folder with child folders

Depending on the current position in the repository

ELO. CreateStructure ("123¶222",0)

Creates a folder with child folders

At the same level as the specified object

ELO. CreateStructure ("123¶222",2345)

Creates a folder with child folders

Within the specified object

ELO. CreateStructure ("123¶222",-764)

1.47 CreateViewer function

With this function, you can create a viewer data record from an export data set. For this purpose, the ViewerPostbox directory must be prepared under ELOprofessional and a completed export data set must be available. The target path may already contain a viewer. If the target repository already exists, the data sets are added to this repository. Up to four repositories may be created in the target area.

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

Example:

```
Set Elo=CreateObject("ELO.professional")
call Elo.CreateViewer("D:\ExportPath", "test1", "D:\Target path")
```

Parameter:

- ExportPath Directory which contains the export dataset
- ArcName Viewer repository name
- DestPath Target directory for the viewer

Return values:

- -1: No active work area
- -2: Target path or repository name is missing
- -3: Directory structure in the target path could not be created
- -4: Intray directory in the target path could not be created
- -5: Viewer files could not be copied
- 1: Ok

Available since: 3.00.354

1.48 DateToInt(AnsiString Date) function

This function converts a date text to an internal ELO numeric date format.

```
int DateToInt( AnsiString Datum )
```

Parameter:

- Text Date to be converted.

Return values:

- 0: Invalid date entry
- >0: ELO date value

Available since: 3.00.196

See also:

- IntToDate

1.49 DebugOut function

This function sends a text to the Elo Debug window.

```
void DebugOut( AnsiString Text )
```

Parameter:

- Text Text to be issued, sent to the Debug window together with the time.

Return values:

- None

1.50 DeleteDocumentVersions function

The command deletes all versions except for the current working version and the milestone versions.

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

Parameter:

- lObjectId: Logical ELO object ID; this must be a document object, folders do not have any versions in any case
- Reserved: Empty (later on, you can add an extension for optional entry of certain versions)

Return values:

- -1: No active work area
- -2: ObjectId does not point to a document
- -3: Error reading database
- -4: No deletion right for document versions
- -5: No read, write or editing rights
- >=0: Number of deleted versions

Available since: 7.00.058

1.51 DeleteMask function

Deletes an ELO keywording form. Before deletion, a check will be performed as to whether there still are entries with this form in the repository or with the deleted but not yet permanently removed objects. If there are still entries, the command is canceled with an error and in the TextParam property, there is a list with the first 16 object IDs that use this form.

```
int DeleteMask( int MaskNo)
```

Parameter:

- MaskNo: Number of the form to be deleted

Return values:

- -1: No active work area
- -2: There are still entries for this form number
- -3: Database error while deleting
- 1: Ok

Example:

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

MaskNo = ELO.LookupMaskName( "ThisFormWillBeDeleted" )
if MaskNo > 0 then
  Res = ELO.DeleteMask( MaskNo )
  if Res > 0 then
    MsgBox "Gelöscht"
  else
    MsgBox "Form could not be deleted, Error number: " & Res & ", Objects: " &
ELO.TextParam
  end if
else
  MsgBox "Form not found"
end if
```

Available since: 4.00.210

See also:

- DeleteObj

1.52 DeleteNote function

The function deletes a sticky note that was accessed beforehand by FindFirstNote or FindNextNote.

```
int DeleteNote()
```

Parameter:

- None

Return values:

- 1: Sticky note was removed
- -1: No active work area
- -2: No sticky note selected
- -3: Delete error
- -4: Insufficient rights for deleting sticky notes
- -5: The sticky note to be deleted is a stamp
- -6: The sticky note to be deleted is currently being edited by another user

Available since: 6.00.090

See also:

- FindFirstNote
- FindNextNote
- UpdateNote

1.53 DeleteObj function

Deletes an ELO entry, determined by the ObjectID. If the target is a folder, all child entries are deleted as well.

```
int DeleteObj( int ObjektId)
```

Parameter:

- ObjectId: Number of entry to be deleted

Return values:

- -1: No active work area
- -2: Error while deleting
- -3: Insufficient right for deletion
- 1: Ok

See also:

- PrepareObj
- UpdateObj

1.54 DeleteProjectOptions function

Deletes all option entries of a project in the defaults for the activities. Any existing project activities are not deleted, however, and can still be displayed and listed.

You can also selectively delete single entries. In this case, instead of the project name, you must enter a string containing name, major no., minor no. and value, separated by the normal delimiter (¶). The field value can be left empty, the major and minor numbers can be set to 0 if they are not known. However, this deletes entire groups, so be careful when using this option.

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

Parameter:

- ProjectName: Name of project or project entry to be deleted

Return values:

- -1: No active work area
- -2: Missing or invalid project name
- -3: Delete error
- -5: Project was not available
- -6: Project entry not properly formatted
- -7: Project entry was not available
- 1: Ok

Available from: 3.00.360

Example:

```
set Elo=CreateObject("ELO.professional")
res = Elo.DeleteProjectOptions( "TEST" )
if res<0 then
  select case res
    case -5
      MsgBox "Project is newly created"
      res=1
    case -8
      MsgBox "You have no authorization for editing the data"
    case else
      MsgBox "Error : " & res
  end select
end if
if res>0 then
  call Elo.InsertProjectOptions( "ELO_SYSTEM", 1, 0, "TEST" )
  call Elo.InsertProjectOptions("TEST" , 10, 1, "Contact")
  call Elo.InsertProjectOptions("TEST" , 10, 4, "Meier")
  call Elo.InsertProjectOptions("TEST" , 10, 2, "Smith")
  call Elo.InsertProjectOptions("TEST" , 10, 3, "Taylor")
end if
```

See also:

- InsertProjectOption

1.55 DeleteSwl function

This function deletes a branch from the keyword list. Every entry of a keyword list is given a unique indicator within its level – a 2-digit letter combination: AA, AB, AC up to ZZ. By concatenating all flags at all levels, you can specifically address every keyword in the tree. Starting with the point for the root, you can now, for example, delete the first entry (AA) below the second (AB) and below that the third (AC) entry using "AAABAC". For this, the selected entry and any existing child entries are removed.

The group designates the assignment of a list to an index field (i.e. the name of the keyword list must be the same as the group field in the keywording forms editor).

```
int DeleteSwl( AnsiString Gruppe, AnsiString Parent)
```

Parameter:

- Group Selects a keyword list
- Parent Path to the entries to be deleted

Return values:

- -1: No open workspace
- -2: Error saving the keyword
- -3: Invalid parent path
- -4: The keyword list is locked by a different user
- 1: Ok

Available from: 5.00.066

Example

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

1.56 DeleteWv function

Deletes a reminder date (determined by the WvIdent parameter). When deleting a date, the owner must also be indicated, if you set a –1 here, your own EloUserId is used.

```
int DeleteWv( int WvId, int UserOwner)
```

Parameter:

- WvId: Number of date to be deleted
- UserOwner: Owner of date

Return values:

- -1: No active work area
- -2: Error while deleting
- 1: Ok

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.57 DeleteWvLine function

With this function, an entry can be hidden within the display of the reminder dates. The associated reminder entry is not deleted.

```
int DeleteWvLine( int LineNo )
```

Parameter:

- LineNo: Line number of the entry to be hidden

Return values:

- -1: No active work area
- 0: Wrong line number
- 1: Ok

See also:

- WriteWv
- DeleteWv
- WvIdent
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.58 DelOutlookName (AnsiString) property

This property indicates the name of the person (group) to be deleted for the reminder. It is used when the name of the person (group) to which the reminder is addressed is changed in ELO so that the entry can be removed.

See also:

- OutlookName

1.59 DoCheckInOut function

This function opens a dialog box to check documents in/out.

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

Parameter:

- hwndParent Windows window handle of the parent window
- DlgTitle Title of dialog box to be opened.
- Short Short name (check-in for new document)
- Desc Additional text (check-in for new document)
- XDate Document date (check-in for new document)
- FileName File name of file
- Ctrl-1 new document
- 0 Check out
- >0 Check in
- Minimize Minimize <>0 for ELO client

Return values:

- -1 No active work area

or

- File to Intray
- 0 Cancel
- >0 Object ID

See also:

- DoCheckInOut2

1.60 DoCheckInOut2 function

This function opens a dialog box to check documents in/out.

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

Parameter:

- hwndParent: Windows window handle of parent window
- sDlgTitle: Title of dialog box to be opened.
- sShort: Short name (check-in for new document)
- sDesc: Additional text (check-in for new document)
- sXDate: Document date (check-in for new document)
- sFilterExt: File endings available for selection. Format expl: "Word documents (*.doc;*.dot;*.txt)|*.doc;*.dot;*.txt|Excel documents (*.xxx)|*.xls;*.txt". Format entry for "All documents|*:*" is entered automatically
- nMinimize: minimize <>0 for ELO client

Return values:

- 0 ... Cancel
- 1 ... New document saved
- 2 ... Document checked in
- 3 ... Document checked out of the repository
- 4 ... Edit document already checked out

1.61 DoCheckInOut3 function

This function opens a dialog box to check documents in/out.

```
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 window handle of parent window
- sDlgTitle: Title of the dialog box to be opened.
- sShort: Short name (check-in for new document)
- sDesc: Additional text (check-in for new document)
- sXDate: Document date (check-in for new document)
- sFilterExt: File endings available for selection. Format example: "Word documents
- (*.doc;*.dot;*.txt)|*.doc;*.dot;*.txt|Excel documents (*.xxx)|*.xls;*.txt".
- Format entry for "All documents|*:*" is entered automatically
- nMinimize: minimize <>0 for ELO client
- iFlags: Bit 0 set: Keywording information will be taken from the internal object, previously initialized with PrepareObjectEx
- Bit 0 not set: Keywording information is extracted from the parameters sShort, sDesc and sXDate
- sInfo: Reserved for later extensions

Return values:

- 0 ... Cancel
- 1 ... New document saved
- 2 ... Document checked in
- 3 ... Document checked out of the repository
- 4 ... Edit document already checked out

1.62 DocId (int) property

The DocId property defines the working version of a document. This entry must have been selected from the list of all document files out of the version history of this document. An foreign entry here can lead to serious malfunctions.

Available since: 5.00.042

See also:

- MaskKey
- DocKey
- DocKind
- DocPath

1.63 DocKey (int) (invalid) property

The DocKey property determines the key of an entry. The entry can be a document or a folder.

If you edit a keywording form definition, this entry contains the default for new documents of this type (the MaskKey contains the key for the form itself).

See also:

- MaskKey
- DocKey
- DocKind
- DocPath

1.64 DocKind (int) property

The DocKind property determines the color of an entry. The entry can be a document or a folder.

If you edit a keywording form definition, this entry contains the default for new documents of this type.

The color number corresponds to the line number from the "System settings -> Font color" dialog box.

See also:

- DocKey
- DocPath

1.65 DocPath (int) property

The DocPath property determines the filing path of an entry. An entry can be a folder or a document.

If the entry is a folder, with the respective system setting, this path can be the default values for documents in this folder. However, this mode only represents a compatibility mode for old ELOoffice repositories, for new repositories it should not be used anymore.

If you edit a keywording form definition, this entry contains the default for new documents of this type.

See also:

- DocKey
- DocKind

1.66 DocTPath (int) property

The DocTPath property determines the default storage path of an entry in a keywording form definition.

See also:

- DocPath

1.67 DoEditObject function

1.68 DoEditObjectEx function

With this function, you can open the keywording dialog box. It displays the data that has been previously initialized via PrepareObjectEx. If the files come from an existing ELO object, this causes conflicts with the automatic display refresh function, which leads to the OLE input data being overwritten by the original data. For this reason, the call must be packed into a CheckUpdate(0) – CheckUpdate(1) bracket.

Requesting DoEditObject corresponds to a DoEditObjectEx(1,-1,0).

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

Parameter:

- Mode: 1 – Edit, 2 – Search
- Handle: Handle of the parent window, default = -1
- Minimize: 1-Minimize ELO main window (only Edit dialog box visible), 0-do not minimize

Return values:

- -1: No active work area
- 1: Dialog box closed with Ok
- 2: Dialog box closed with Cancel

Example:

```
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
```

1.69 DoExecute function

With this function, another application can be started; internally, the Windows API function *ShellExecute* is requested.

Option 1: The name of a program (EXE file) is transferred in the FileName parameter.

Option 2: In the FileName parameter, the name of a file is transferred, the application associated with this file type is started.

```
int DoExecute(AnsiString FileName)
```

Parameter:

- FileName File to be opened/run

Return values:

- <=32: Windows error code of the *ShellExecute* function
- >32 : Success

See also:

- RunEloScript
- DoExecuteEx

1.70 DoExecuteEx function

With this function, another application can be started; internally, the Windows API function *ShellExecute* is requested.

Option 1: The name of a program (EXE file) is transferred in the FileName parameter.

Option 2: In the FileName parameter, the name of a file is transferred, the application associated with this file type is started.

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

Parameter:

- File File to be opened/run
- Param Additional parameters (may remain empty)
- Directory Start directory (may remain empty)
- Action "Open", "Print" or "Explore", (may remain empty, then "Open" is run)
- Mode Window size upon start (-1: SW_SHOWNORMAL). The possible constants can be found in the Windows help under "ShellExecute".

Return values:

- <=32: Windows error code of the *ShellExecute* function
- >32 : Success

See also:

- RunEloScript
- DoExecute

1.71 DoFullTextSearch function

The DoFullTextSearch function initiates a full text search. As result of the function, the number of references found is returned, these are then available in the search results list and can be displayed via SelectLine.

```
int DoFullTextSearch(AnsiString sSearchString, int iSearchOption)
```

Parameter:

- sSearchString Text to be found
- iSearchOption 0=AND-OR-linked search terms
- 1=intuitive search

Return values:

- -1: No active work area
- -2: Error during full text search
- else: Number of found entries.

See also:

- DoSearch
- SelectLine

1.72 DoInvisibleSearch function

The DoInvisibleSearch function starts an invisible search process from the ObjShort, ObjMemo, and ObjAttrib entries. For this purpose, first of all a new object dataset is created with PrepareObject, this dataset is fed with the terms to be found via the access operations and the DoInvisibleSearch function is requested. As result of the function, the number of found references is returned, these are then available in an invisible internal search results list and can be requested via getEntryId(). However, for this purpose a special line number must be transferred to the getEntryId function, so that it can detect that it is not the normal list, but the invisible list to be searched. This number consists of the line number and a constant factor of 268435456 (that is 0x10000000).

```
int DoInvisibleSearch()
```

Parameter:

- None

Return values:

- -1: No active work area
- else: Number of entries found.

Available from: 3.00.188

Example:

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

'suche alle Rechnungseinträge
MaskNo=Elo.LookupMaskName("Invoice")
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
```

See also:

- PrepareObject
- SelectLine
- DoFulltextSearch

1.73 DoSearch/DoSearchSel(AnsiString)/DoSearchEx(AnsiString, int) function

The DoSearch function starts a search process from the ObjShort, ObjMemo, and ObjAttrib entries. For this purpose, first of all a new object dataset is created with PrepareObject, this dataset is fed with the terms to be found via the access operations and the DoSearch function is requested. As result of the function, the number of references found is returned, these are then available in the search results list and can be displayed via SelectLine.

Another special mode is available for displaying a list of ELO objects. In this case a keywording form of the type "Basic entry" is entered so that only the field with the short name is entered with a string of the type "¶Id1¶Id2¶Id3¶ ... ¶IdN¶". A list with these objects is then displayed in the Search work area.

You can determine via the mode parameter if an already existing search result is to be deleted before a search. You can then create multiple searches with this parameter. For the first search a deletion is carried out, all subsequent searches then supplement the list created so far.

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

Parameter:

- SelectObject If the search result contains an entry with the predefined short name, this line is selected.
- Mode Bit 0: 0 do not delete old list content 1: delete before searching
- Bit1..31: reserved.

Return values:

- -2: General SQL error
- -1: No active work area
- else: Number of entries found.

Available since: DoSearchEx: 3.00.358

Set ObjFlags: Bit 29: 1 The version history is searched during the search

Restrict search by object type:

Only search for documents Elo.ObjTypeEx = 9998

Only search for folders: Elo.ObjTypeEx = 9999

Example:

```
' carries out three searches for the terms ELO, test and word
' and shows the result in a results list
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 )
```

See also:

- PrepareObject
- SelectLine
- DoFulltextSearch

1.74 DoSelArcTree function

1.75 DoSelArcTreeEx function

1.76 DoSelArcTree3 function

Opens a dialog box for selecting an ELO entry.

```
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 of the parent window or ZERO
- sDlgTitle: Dialog box title
- nCtrl: 1 to specify that only folders can be selected
- nMinimize: <> 0 to minimize ELO.
- ObjId: DoSelArcTreeEx opens the tree at the transferred ObjectID
- nRoot: DoSelarcTree3 shows a partial tree from the transferred ObjectID (nRoot) and opens this partial tree at the transferred ObjectID (ObjId).

Return values:

- -1: for errors,
- 0: for cancel,
- >0: Object ID of selected entry

1.77 EditActivity (AnsiString) function

This function edits an activity. You can assign defaults via the input string. In particular, if you create a new activity for a document, you must initialize the EloGuid with the GUID of the document. If possible, you should also assign defaults for the project entry.

The ActInfo string consists of the following parts:

```
ActGuid          Unique ID of activity; leave empty for new entries!
DocGuid     GUID of ELO document (determine with GetGuidFromObj).
Destination Recipient
RevVers          Version
ActTStamp   Timestamp, leave empty; automatically generated when saved
Project          Project name, assign defaults if possible
Owner       Owner, ELO user number
Creator          Creator, ELO user number
Prio        0,1 or 2
ShortDesc   Short description
SentAt           Send date in ISO format
SentMode    Send mode
DueDate     Expected return date, ISO format
BackAt           Return date, ISO format
BackMode    Return status
Comment     Comment
FileName    Name of file sent
UD0   User-defined field 1
UD1
...
UD9   User-defined field 10
```

```
AnsiString EditActivity( AnsiString ActInfo )
```

Parameter:

- ActInfo: Current field allocation, the individual texts are linked by the (¶) separator.

Return value:

- Empty An error has occurred or the user has closed the dialog box with "Cancel".

Available since: 3.00.360

Example:

```
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
```

1.78 EditDlgActive (int) property

This property indicates if a dialog box has been opened from ELO.

Return values:

- 0 – No
- 1 – Yes

1.79 EditWv (int WvId, int ParentId) function

This function edits a reminder date.

```
int EditWv (int WvId, int ParentId)
```

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

Return value:

- -1 No active work area
- 0 Dialog box closed with Cancel
- 1 Dialog box closed with OK
-

See also:

- WVNew
- WvDueDate
- WvListInvalidate
- WvActionCode

1.80 EloWindow function

With this function, you can determine how the ELO work space is displayed. 'MINIMIZE' can be used as a parameter, in this case, no work interface can be seen. With 'MAXIMIZE', the work interfaces take up the full screen or with 'NORMAL', that is, the default settings are used for the size of the work interface.

```
int EloWindow( AnsiString DisplayMode )
```

Parameter:

- DisplayMode 'MINIMIZE', 'NORMAL' or 'MAXIMIZE'

Return values:

- 0,1,2: Number of open work areas

See also:

- BringToFront

1.81 Export function

You can export a (partial) repository with the export function.

The ID for the available export options can be found in the following list:

```
// Do not copy documents from the file structure
#define IEX_SUPPRESSDOCS      (1 << 0)
// Copy chaos docs
#define IEX_CHAOSDOCS        (1 << 1)
// Delete copied documents after completion
#define IEX_ERASEDOCS         (1 << 2)
// Create separate file structure when importing
#define IEX_NEWHIRARCHY       (1 << 3)
// Use filing date instead of document date
#define IEX_USEIDATE          (1 << 4)
// Supply complete storage information
#define IEX_PATHINFO          (1 << 5)
// Export repository entry
#define IEX_ARCHIVEENTRY      (1 << 7)
#define IEX_TXT_ARCHIVEENTRY "Repository Entry"
// Export clipboard
#define IEX_CLIPBOARD         (1 << 8)
#define IEX_TXT_CLIPBOARD     "Clipboard"
// Export searches
#define IEX_SEARCHLIST        (1 << 9)
#define IEX_TXT_SEARCHLIST    "Searchlist"
// Export encrypted documents
#define IEX_EXPORT_CRYPTED   (1 << 10)
// Document/Attachment versions
#define IEX_EXPORT_DOCVERS   (1 << 11)
#define IEX_EXPORT_ATTVERS   (1 << 12)
// What happens during the export of keyworded documents without a valid
password
#define IEX_EXPORT_CRYPT_QUERYPASSWD   (1 << 13)
#define IEX_EXPORT_CRYPT_SKIP   (1 << 14)
// Replace references with originals
#define IEX_REFS_AS_ORGS   (1 << 15)
// Export keyword lists
#define IEX_BUZZWORDS   (1 << 16)
```

Application:

(1 << 0) = 2^0 (Bit 0 set) 1

(1 << 1) = 2^1 (Bit 1 set) 2

(1 << 2) = 2^2 (Bit 2 set) 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“ )
```

Return

- -1 No active work area
- -2 Error during export
- -3 Target path could not be created
- 1 OK

Please note that the export writes a report file which you should use for evaluating possible errors in the Intray of the active user.

1.82 FindFirstNote function

You can access the sticky notes of a document with the help of the FindFirstNote and FindNextNote functions. The keywording information must be entered using PrepareObjectEx before running FindFirstNode.

```
int FindFirstNote()
```

Parameter:

- -

Return values:

1: Sticky note was found

0: No sticky note found

Available since: 6.00.090

Example:

Determine and display all sticky notes for the document with the ObjectID iID.

```
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 & "Sticky note no." & iCount & ": " & Elo.NoteText & vbCRLF & vbCRLF
  iRes=Elo.FindNextNote
Loop
MsgBox sTxt
```

See also:

- FindNextNote
- UpdateNote
- DeleteNote

1.83 FindFirstWv function

This function is used if reminder dates need to be determined for a certain ELO object. The function contains the object ID of the desired ELO object as parameter; it returns the number of the reminder dates that are found. With the FindNextWv function, the IDs of the reminder dates are checked out.

```
int FindFirstWv( int ObjektId )
```

Parameter:

ObjectId ID of the ELO object

Return values:

>=0: Number of reminder dates found

-1: No active work area

-2: Error while reading the reminder dates

See also:

- FindNextWv

1.84 FindNextWv function

With this function, the reminder dates associated with an ELO object are checked out, first of all FindFirstWv must be requested.

```
int FindNextWv()
```

Parameter:

- None

Return values:

- >=0: ID of the reminder
- -1: no reminder date available

See also:

- FindFirstWv

1.85 Function FindUser/FindUserEx function

This function searches the internal ELO user number for a name.

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

Parameter:

- UserName Name of user for which the user ID is to be found.
- Mode 0: ELO Name, 1: NT Name, 2: Outlook Name.

Return values:

- UserID or -1 if no user was found with this name

Available since: 5.00.162 for FindUserEx

See also:

- ActiveUserId
- LoadUserName
- SelectUser

1.86 FreezeDoc/FreezeDocEx function

The FreezeDoc function enables you to convert an ELO document to a Tiff file via the Tiff printer and attach it to the document as a new version.

There are a few requirements you need to meet in advance:

The ELO Tiff printer must be registered in the options.

The print temp directory must be empty before executing the command, which you can do with UpdatePostbox, for example.

During conversion, no other interfering print commands may be activated.

Conversion is carried out via a ShellExecute ("print" ...) request. For this reason, the respective application must be installed for the document on the client computer. In addition, some applications (depending on the document type/content) open their own print dialogs.

Common problems when printing:

(Outlook): As long as Outlook is open, it does not respond to changes to the default printer, meaning the command to switch to the Tiff printer is ignored. For this reason, close all open Outlook windows before converting MSG files. Alternatively, you can configure the Tiff printer as the default printer in your system.

(PowerPoint): When converting PPT files, a separate print dialog box is displayed that must be acknowledged manually by the user.

Printing is only possible in Windows NT, Windows 2000 and Windows XP, as there are too many functional limitations in Windows 95/98/ME.

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

Parameter:

- ObjectId Logical number of the document to be converted
- iPrinter: Logical number for the printer
- 1 = TIFF conversion
- 2 = PDF conversion

Return values:

- -1: No active work area
- -2: Error while printing
- 1: Ok

Example:

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

if Elo.SelectView(0)<>1 then
  MsgBox "This script can only be executed in the Repository work area."
```

```
else
  ' search through all documents in current folder
  for i=0 to 10000
    id=Elo.GetEntryId(i)
    if id<1 then exit for

    ' check whether it is a document
    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
```

Available since: 3.00.348

1.87 FromClipboard function

This function copies a text from the Windows Clipboard.

```
AnsiString FromClipboard()
```

Parameter:

- None

Return values:

- Text from the Windows Clipboard

Available since: 3.00.456

See also:

- ToClipboard

1.88 GetArcKey function

This function determines the current repository key.

```
int GetArcKey(  )
```

Parameter:

- None

Return values:

- Repository key or -1 for error

1.89 GetArcName function

This function determines the current repository name.

```
AnsiString GetArcName(  )
```

Parameter:

- None

Return values:

- Repository name or empty string in case of error

1.90 GetArchiveName function

With the GetArchiveName function, you can determine the repository name for a given repository number. Via the repository number –1, you receive a list of all available repositories, separated by the system separator (usually ¶).

```
AnsiString GetArchiveName( int ArchiveNo )
```

Parameter:

- ArchiveNo: -1: A list of all available repositories
- 0..n: Name for given repository number, empty if invalid

Return values:

- ArchivName or empty in case of error

1.91 GetAutoDlgValue (int Index) function

Supplies the entered or selected value of the dialog box objects.

```
AnsiString GetAutoDlgValue (int Index)
```

Index : Object position, starting with 1 for the first object.

Return value:

For check boxes and radio buttons 0 – Unchecked 1 – Checked

For Edit fields the content of the input field is returned.

For labels an empty string is returned.

Example:

```
' Returns the text of the input field and checks/unchecks the check box.
Elo.CreateAutoDlg ("Personal info")
Elo.AddAutoDlgControl (4,1,"Name","")
Elo.AddAutoDlgControl (2,3,"Married? Yes","0")
Elo.ShowAutoDlg
MsgBox GetAutoDlgValue (1)
MsgBox GetAutoDlgValue (2)
```

Available from: Version 3.00.228

See also:

- CreateAutoDlg
- AddAutoDlgControl
- ShowAutoDlg

1.92 GetBarcode function

With this function, barcodes that have previously been determined with the ReadBarcodes function are read.

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

Parameter:

- Index Barcode index

Return values:

- <>"": Barcode
- "": Error (e.g. wrong index, ReadBarcodes not run previously)

See also:

- ReadBarcodes

1.93 GetChildNode function

With this function the successor nodes of the current node are checked out within a process in a loop.

```
int GetChildNode()
```

Return values:

- -2: no more successor nodes available
- -1: no node activated
- 1: Ok

See also:

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

1.94 GetCookie function

The GetCookie function reads the value of a cookie entry and returns it to the requesting program. This access is primarily intended so that ELO scripting macros can permanently store information in ELO. The cookie memory is deleted when ELO is closed.

You access a cookie with its name (Ident); the value assigned to it is returned. If the cookie is unknown, an empty string is returned.

Please note that the number of cookies is limited. Each macro and any other external program should only use a few of them and each time you launch it you should use the same ones again and not keep creating new ones.

```
AnsiString GetCookie( AnsiString Ident )
```

Parameter:

- Ident Cookie name, was set in SetCookie

Return values:

- "": Unknown or empty cookie
- else: Content of requested cookie.

See also:

- SetCookie

1.95 GetCounter function

The GetCounter function supplies the current status of the selected counter. If the Increment parameter contains a value other than 0, the counter automatically counts upwards.

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

Parameter:

- CounterNameName of counter to be triggered
- Increment 0: only determine current counter status
- 1: Determine counter status and count upwards

Return values:

- -1: Error
- else: counter status

See also:

- GetCounterList
- CreateCounter

1.96 GetCounterList function

The GetCounterList function supplies a string with all AccessManager counters that are currently logged into the system. The individual entries are separated by a ¶.

```
AnsiString GetCounterList()
```

Parameter:

- None

Return values:

- "": Error while reading the counter list
- else: counter list.

See also:

- GetCounter
- CreateCounter

1.97 GetDocExt function

Returns the extension for an object or document ID (Windows document type).

```
AnsiString GetDocExt( int ObjDocId, int Status )
```

Parameter:

- ObjDocId Internal ELO object or document ID
- Status Bit 0: 0: DocumentId, 1: ObjectId

Return values:

- Extension of the document.

Available since: 3.00.392

Example:

```
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 "Document type for entry '" & Elo.ObjShort & "' : " & Ext
  end if
end if
```

See also:

- UpdateDocument
- InsertDocAttachment
- GetDocumentPathVersion

1.98 GetDocFromObj function

This function acquires the document manager Document ID of a logical Object ID. This ID can be used for document-specific actions (e.g. GetDocumentPathName or GetDocumentSize).

The Flags parameter can be used to define if the current working version of the document or of the file attachment is returned:

- Return current document version (working version)

Return current attachment

```
int GetDocFromObj(int ObjId, int Flags )
```

Parameter:

- ObjId: Logical Object ID of the entry to be searched
- Flags: 0 or 1

Return values:

- -1: No active work area
- -2: ObjID not found
- -3: Invalid object
- 0: No document file assigned
- >1: Ok, Document ID

Available since: 5.00.018

Example:

```
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
```

1.99 GetDocRefComment function

With this function you can request the version entry and comment for a document version. The two fields are returned in a string, separated by the pipe symbol "|". Please note that only two fields are returned at present, this may change in future. You must keep in mind in your scripts that any number of fields may be added.

```
AnsiString GetDocRefComment( int ObjectId, int DocumentId )
```

Parameter:

- ObjectId ELO ObjectId of the document
- DocumentId Access Manager DocumentId of the document version

Return values:

- Comment type+version text, empty for error.

Available since: 3.00.322

Example:

```
set Elo=CreateObject( "ELO.professional" )
ObjId=Elo.GetEntryId(-1)
if ObjId>0 then
  DocId=Elo.GetDocumentPathVersion( ObjId, 10, 0 )
  if DocId=0 then
    MsgBox "This is not a document or" & vbcrlf & "the document has no
attachment"
  else
    DocInfos=Split(Elo.GetDocRefComment( ObjId + 1073741824, DocId ),"|")
    MsgBox DocInfos(1)
  end if
else
  MsgBox "No document is active"
end if
```

See also:

- GetDocumentPathVersion

1.100 GetDocumentExt function

Returns the file ID (extension). The Document Manager ID must and not the logical Object ID must be specified as the DocID. You can get this number with the GetDocFromObj. request, for example.

```
AnsiString GetDocumentExt( int DocId )
```

Parameter:

- DocId Document Manager file Id

Return values:

- Extension or empty in the case of an error

Available since: 5.00.018

Example:

```
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
```

1.101 GetDocumentOrientation function

With this function, you can have an image file in the Intray analyzed in order to detect a possible rotation by 90, 180 or 270 degrees. The analysis is performed using the OCR system. Before using the function, you need to run the OCRInit function, and the OCRExit function once it has ended. With the RotateFile function, the file can be rotated after analysis so that texts can be read from the bottom (only single-page TIFF files).

As a file name, you can use the entry from the Intray (without the path, only the file name) or an index in the list of Intray documents (indicated by a # sign, e.g. #0 is the first entry in the list.)

```
int GetDocumentOrientation( AnsiString FileName,  int PageNumber )
```

Parameter:

- FileName: Name of the file to be examined
- PageNumber: Page number (page 1 = "0")

Return values:

- -1: No active work area
- -2: Error loading the file
- -3: Error when detecting rotation
- 1..4: Orientation (1=0°, 2=90°, 3=180°, 4=270°)

See also:

- OrientFile
- RotateFile
- UpdatePostbox

1.102 GetDocumentPath function

Reads the document or attachment file from a document and returns an access path to the file. Normal repository documents are read-only, for this reason the reader can request a free copy via the status (AUTO_WRITEACCESS). However, the copy is only valid for a limited period and is deleted automatically when closing ELO.

```
AnsiString GetDocumentPath( int DocId, int Status )
```

Parameter:

- DocID Internal ELO ObjectID
- Status Bit 0: Write access to copy
- Bit 1: Attachment instead of document file
- Bit 2: Full text representation instead of document file
- Bit 3: List of DocumentIDs instead of file path
- Bit 4: Signature file instead of document file
- The bits 1,2,3 and 4 cancel each other out and may not be combined.

Return values:

- Access path to document or file attachment

See also:

- UpdateDocument
- InsertDocAttachment
- GetDocumentPathVersion

1.103 GetDocumentPathName function

Returns the file path of a document file number. This file path is built from the view of the document manager and is generally not accessible from the Client. The Document Manager ID and not the logical Object ID must be specified as DocID. You can get this number with the GetDocFromObj. request, for example.

```
AnsiString GetDocumentPathName( int DocId )
```

Parameter:

- DocID Document manager file ID

Return values:

- Access path to document or file attachment or empty in the case of error

Available since: 5.00.018

Example:

```
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
```

1.104 GetDocumentPathVersion function

Reads the document or attachment file from a document and returns an access path to the file. For version-controlled documents, previous versions can be accessed via the *Version* parameter. For this purpose, *Version* is increased by 1 starting with 0 (=current version) until an empty string is returned.

Normal repository documents are read-only, for this reason the reader can request a free copy via the status (AUTO_WRITEACCESS). However, the copy is only valid for a limited period and is deleted automatically when closing ELO.

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

Parameter:

- DocID Internal ELO ObjectID
- Status Bit 0: Write access to copy
- Bit 1: Attachment instead of document file
- Bit 2: Full text representation instead of document
- Bit 3: Document numbers instead of document

Version

- 0 = current version
- 1... = versions
- If bit 3 is set from the status, 0 only returns the current
- number, 1 returns all numbers

Return values:

- Access path to document or file attachment
- If bit 3 is set from the status, the return value contains one or more
- numbers, separated by a pipe symbol numbers, separated by a pipe icon "|".

Available since: 3.00.322 for the list (bit 3 of the status field)

Example:

```
ObjId=Elo.GetEntryId(-1)
if ObjId>0 then
  DocIds=Elo.GetDocumentPathVersion( ObjId, 10, 1 )
  if DocIds="" then
    MsgBox "This is not a document or" & vbcrlf & "the document has no
attachment"
  else
    MsgBox DocIds
  end if
else
  MsgBox "No document is active"
end if
```

See also:

- UpdateDocument
- InsertDocAttachment

- GetDocumentPath

1.105 GetDocumentSize function

Returns the file size for a document file number. The Document Manager ID and not the logical Object ID must be specified as the DocID. You can get this number with the GetDocFromObj. request, for example.

```
Int  GetDocumentSize( int DocId )
```

Parameter:

- DocID Document manager file ID

Return values:

- -1: No active work area
- -2: Error while reading the document manager information
- >=0: File size

Available since: 5.00.018

Example:

```
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
```

1.106 GetEntryId function

This function returns the internal ELO object ID of a line from the Repository, Intray or Search work area. This way, you can receive a list of all visible objects on the left side by calling the GetEntryId function, starting with line 0 and increasing until a 0 is returned.

If you start the request in the reminder, you receive a ReminderID instead of an ObjectID as long as the line is a reminder entry. If the line addressed is a workflow entry, you are given the ObjectID of the repository element that is in the workflow. To differentiate between workflows and reminders, you can perform a check after calling up GetEntryID with the help of the IsWFLine property.

When requesting the function, there are a few "special" lines:

- -1: The currently selected entry is read
- -2: The number of the internal EloObject is read
- -10: The selected line is read from the top-level folder list
- -11: The selected line is read from the 2nd-level folder list
- -12: The selected line is read from the 3rd-level folder list
- -13: The selected line is read from the document list

```
int GetEntryId( int Line )
```

Parameters

- Line Line to be read (0…) or -1 … -13

Return values:

- Internal ELO ObjectID
- 0: (error, no additional entry, no selection performed)
- -1: No active work area
- See also:

  - GetEntryName
  - IsWFLine
  - ReadWFNode

1.107 GetEntryName function

With this function, you can quickly determine the short name of an ELO object via the internal ELO object number. If you use the value 0 as ObjectID, you receive the short name of the currently selected entry.

```
AnsiString GetEntryName( int ObjId )
```

Parameter:

- ObjID ELO ObjectID or 0 for the currently selected entry

Return values:

- Short name or an empty string if there is an error

See also:

- GetEntryId

1.108 GetGuidFromObj function

This function determines the ELO GUID for a given ELO ObjectID.

```
AnsiString GetGuidFromObj( long ObjId )
```

Parameter:

- GUID GUID of the object to be found

Return values:

- Empty: no work area active or object not found
- Else: GUID

Available from: 3.00.176

See also:

- ObjGuid
- GetObjFromGuid

1.109 GetHistDoc function

Returns the internal DocumentID for the n-th hit from LookupHistMD5.

```
int GetHistDoc (int HitNumber )
```

Parameters

- HitNumber: 0..n (n is returned by LookupHistMD5)

Return values:

- -1: Faulty HitNumber
- >0: DocumentID

Available since: 3.00.170

See also:

- LookupHistMD5
- GetHistObj
- GetMD5Hash

1.110 GetHistObj function

Returns the internal DocumentID for the n-th hit from LookupHistMD5.

int GetHistObj (int HitNumber)

Parameters

- HitNumber: 0..n (n is returned by LookupHistMD5)

Return values:

- -1: Faulty HitNumber
- >0: ObjectID

Available since: 3.00.170

See also:

- LookupHistMD5
- GetMD5Hash
- GetHistDoc

1.111 GetIndexGroups function

With the GetIndexGroups function, you can determine a list of values entered for an index line, duplicates are ignored. If you have an "Article name" index field, for example, you can create a list of all available article names via GetIndexGroups. You can then use these for selection in a combo box. Please note that this function can only be used appropriately if the amount of hits does not exceed several dozen to several hundred entries.

This function can be used in two different ways. Only one index field is considered for the direct list (article example). However, you can also create an "indirect" list, e.g. all article names that have been obtained from the customer with the customer number (CTNO) 4711. This concerns two index fields, first of all one that determines the selection and another that contains the items to be listed. However, it needs to be noted that a considerable database load is created for a large number of hits.

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

Parameter:

- GroupCol: Group name of index field from which the list is to be created
- SelCol: Group name of index field from which the selection is to be used
- SelVal: Index limit

Return values:

- Hit list or empty if there is an error

Available since: 3.00.324

Example:

A list is created from index line ELOFAQBER with all the different entries that contain the string P3. in the ELOVER index field.

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

1.112 GetLastDocId function

This function determines the last physical document number in the repository.

```
Int GetLastDocId()
```

Parameter:

- none.

Return values:

- -1: No active work area
- -2: Document number could not be determined.
- >0: Last DocumentID

Available from: 3.00.170

See also:

- Backup

1.113 GetLastVersionTimeStamp (int, int) function

Returns the filing date of the physical document file.

```
int GetLastVersionTimeStamp( int DocId, int Status )
```

Parameters

- DocID: ELO object ID of the document
- Status: 0: current document, >1 version
- 0x1000 added as additional flag: Attachment instead of document

Return values:

- -1: No active work area
- -2: No document found
- -3: Invalid status
- -4: Document data could not be read from the database
- >0: date in internal ELO format

Available since: 3.00.170

See also:

- IntToDate

1.114 GetListEntry function

With this function, you can check out a line from the internal hit list of CollectWv and DoInvisibleSearch. The result consists of the following parts:

Type¶ObjID¶Owner¶ID1¶ID2¶Text

- Type AC: activity, OB: object, WF: workflow task, WV: Reminder date
- ObjID ELO ObjectID
- Owner Owner of the date (always 0 for OB)
- ID1 OB: attachment ID, WF: workflow Id, else 0
- ID2 WF: node Id, else 0

```
AnsiString GetListEntry( int Index )
```

Parameter:

- Index Line number 0..(number of hits –1)

Return values:

- Empty: No work area active or line not available
- Else: Line content

Available since: 3.00.394

Example:

```
Set Elo=CreateObject("ELO.professional")
UserId=Elo.ActiveUserId
' nur die eigenen Termine
'UserID=Elo.ActiveUserID + 1073741824 ' with group deadlines
'UserID=Elo.ActiveUserID + 536870912 ' with substitutes
'UserID=Elo.ActiveUserID + 1610612736 ' all deadlines

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
```

See also:

- CollectWv
- DoInvisibleSearch

1.115 GetMaskLineAcl function

When editing a form definition, the "right" of an index field can be queried with the GetMaskLineAcl function.

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

Parameter:

- iKeyNo Index field number (0 ... 49)

Return values:

- - : Empty string for unknown 'iKeyNo'
- otherwise the ACL of the index field

Starting with Client version 8.00.056

See also:

- SetMaskLineAcl
- ReadObjMask
- WriteObjMask

1.116 GetMD5Hash function

With this function, you can determine the MD5 hash for a file or for a data block. If you provide a file name as a parameter, you receive the hash for the file. If you enter a string starting with the "##" symbols, you will be provided with a hash value for this string (the ## symbols themselves are not counted for this)

```
AnsiString GetMD5Hash( AnsiString FileName )
```

Parameters

- FileName File name or input string for the MD5 hash value

Return value:

- MD5 hash value as 32-character HEX string

Available from: 3.00.170

See also:

- LookupHistMD5
- GetHistObj
- GetHistDoc

1.117 GetObjAttrib function

This function checks out the value of an input line of the current document mask.

There are three values for each input field:

The "visible" description, shown to the user in the keywording forms as a field name (e.g. invoice number). It provides a line name for the user.

The extension, which is used in the database to identify the line in the index (e.g. RENR). It provides a line reference for the machine.

The input value; this stores the user input (e.g. 199807106)

```
AnsiString GetObjAttrib( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Current content of the input value

See also:

- SetObjAttrib
- GetObjAttribKey
- GetObjAttribName
- GetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

1.118 GetObjAttribFlags function

This function reads out the flags from an input field of the current keywording form. The return value is an integer, the bits set here mean the following:

- Bit 0 Only accept data from keyword list
- Bit 1 Automatically add * before search term
- Bit 2 Automatically add * after search term
- Bit 3 New tab after this line
- Bit 4 Line invisible
- Bit 5 Read-only line
- Bit 6 Column with high priority, add text to the short name

```
int GetObjAttribFlags( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Current flags of the keywording form addressed

Available from: 5.00.164

See also:

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

1.119 GetObjAttribKey function

This function reads out the index description from an input field of the current keywording form.

There are three values for each input field:

- The "visible" description, shown to the user in the keywording forms as a field name (e.g. invoice number). It provides a line name for the user.
- The extension, which is used in the database to identify the line in the index (e.g. RENR). It provides a line reference for the machine.
- The input value; this stores the user input (e.g. 199807106)

```
AnsiString GetObjAttribKey( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Current content of the index name field

See also:

- GetObjAttrib
- SetObjAttribKey
- GetObjAttribName
- GetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

1.120 GetObjAttribMax function

This function reads the maximum entry length of an input field for the current keywording form. This field is evaluated in the default input form within ELO. If you make your inputs using the OLE Automation program, you need to keep the maximum and minimum input lengths in mind.

```
int GetObjAttribMax( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Maximum length of the entry, 0: no control, any length

See also:

- GetObjAttrib
- GetObjAttribKey
- GetObjAttribName
- GetObjAttribFlags
- GetObjAttribMin
- SetObjAttribMax
- GetObjAttribType

1.121 GetObjAttribMin function

This function reads the minimum entry length of an input field for the current keywording form. This field is evaluated in the default input form within ELO. If you make your inputs using the OLE Automation program, you need to keep the maximum and minimum input lengths in mind.

```
int GetObjAttribMin( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Minimum length of the entry, 0: no control, any length

See also:

- GetObjAttrib
- GetObjAttribKey
- GetObjAttribName
- GetObjAttribFlags
- SetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

1.122 GetObjAttribName function

This function reads out the name from an input field of the current keywording form.

There are three values for each input field:

- The "visible" description, shown to the user in the keywording forms as a field name (e.g. invoice number). It provides a line name for the user.
- The extension, which is used in the database to identify the line in the index (e.g. RENR). It provides a line reference for the machine.
- The input value; this stores the user input (e.g. 199807106)

```
AnsiString GetObjAttribName( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Current content of the name field

See also:

- GetObjAttrib
- GetObjAttribKey
- SetObjAttribName
- GetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- GetObjAttribType

1.123 GetObjAttribType function

This function reads the type of input of an input field of the current keywording form. This field is evaluated in the default input form within ELO. If you make your inputs using the OLE Automation program, you need to keep the type of entry in mind.

- 0: any text field
- 1: Date field
- 2: Numerical field
- 3: File number
- 4: ISO date
- 5: List entry
- 6: User field
- 7: Thesaurus
- 8: Numeric, fixed width
- 9: Numeric, fixed width, 1 decimal place
- 10: Numeric, fixed width, 2 decimal places
- 11: Numeric, fixed width, 4 decimal places
- 12: Numeric, fixed width, 6 decimal places

```
int GetObjAttribType( int AttribNo )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- line 51 contains the file names of the document

Return values:

- Type of entry permitted

See also:

- GetObjAttrib
- GetObjAttribKey
- GetObjAttribName
- GetObjAttribFlags
- GetObjAttribMin
- GetObjAttribMax
- SetObjAttribType

1.124 GetObjFromDoc function

This function determines the object associated with a certain DocumentID (the DocumentID belongs to the file, not the keywording – this ID is currently being determined here).

```
int  GetObjFromDoc( int DocId)
```

Parameter:

- DocID File number of the source document

Return values:

- -1 No active work area
- 0 DocID not found, no keywording assigned
- >0 ObjectID of keywording for the file

1.125 GetObjFromDocEx function

This function determines the associated documents for an AccessManager document ID. In contrast to the GetObjFromDoc function, here not only the current working document but also the attachment as well as the document and attachment versions is considered. The decision which parts are to be considered by the search is controlled by the flags parameter:

- 0 Only consider the current document version (working version)
- 1 Only consider the current attachment
- 2 Consider all document versions
- 3 Consider all attachment versions
- 4 Consider all document and attachment versions

```
int GetObjFromDocEx(int DocId, int Flags )
```

Parameter:

- DocID: AccessManager document number of the entry to be searched
- Flags: 0..4

Return values:

- -1: No active work area
- -3: DocID not found
- >1: Ok, ObjectID

Available since: 4.00.106

Example

```
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
```

1.126 GetObjFromGuid function

This function determines the ELO GUID for a given ELO ObjectID.

```
int GetObjFromGuid(AnsiString Guid)
```

Parameter:

- GUID GUID of the object to be found

Return values:

- -1 No active work area
- GUID not found
- >0: ObjectID

Available from: 3.00.176

See also:

- GetGuidFromObj
- ObjGuid

1.127 GetObjMaskNo function

Returns the number of the current form (document type). An equivalent SetObjMaskNo does not exist for this; for new ELO documents, the document type is set "forever" once with the PrepareObject.

```
int GetObjMaskNo()
```

Parameter:

- None

Return values:

- Number of the current keywording form

See also:

- ReadObjMask
- WriteObjMask

1.128 GetObjRef (int ObjId, int RefNo) function

In addition to its hierarchical tree structure (cabinet - folder - tab - document), ELO also lets you create other references. A document called "Miller invoice" can be filed into the "Invoices" folder, while an additional reference is created in the "Miller" folder. Although there then is only one instance of that document in the system, it can be viewed and processed in both places.

You can determine the path of the reference with this function.

```
AnsiString GetObjRef (int ObjId, int RefNo)
```

- ObjId : Internal ELO ID
- RefNo : Number of the reference to be displayed (starting with 0)
- (0 is the own reference)

Return value:

- -1 - No active work area
- -2 - Reference not available
- Otherwise, the path for the reference, separated with the separator

Example:

- Returns the path of the second reference from the object with ID 245.
- Ref2 = Elo.GetObjRef (245,2)

See also:

- InsertRef
- RemoveRef

1.129 GetOcrRectList function

This function returns a list of hit rectangles for a searched word.

There is a list of the hit positions [xStart, yStart, xEnde, yEnde][…] in RectList

If you are looking for different terms, then the actual OCR process is only performed once, but the text is stored in an internal cache in the Client. For this reason, "FileName" may also only contain a valid ELO repository file name "123456789.tif". An Intray file must be renamed accordingly and the DocId must be long enough that it cannot be confused with the actual documents (such as starting with 1000000). The hit cache is deleted every time the client is started.

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

Parameter:

- FileName: Name of the file to be examined This must be a name in the ELO repository date format and it must be unique.
- PageNumber: Page number of the Tiff document to be searched.
- SearchText Searched term
- Flags Bit 0: Keep uppercase and lowercase spelling in mind

Return values:

- List of the hit positions [xStart, yStart, xEnde, yEnde][…]

Example:

```
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
```

1.130 GetPopupObjectId() function

With this function, you can determine the object currently selected for a context menu. This entry is only valid when it was requested as a result of a context menu event.

```
int GetPopupObjectId( )
```

Parameter:

- None

Return values:

- -1: no active work area
- >0: current ObjectId

Available from: 3.00.196

1.131 GetPostDir function

With this function, you can quickly determine the Intray path of the current user.

```
AnsiString GetPostDir( )
```

Parameter:

- None

Return values:

- Intray path or an empty string in the case of an error

1.132 GetRegInfo function

With this function, the serial number information of the Access Manager can be requested. For this purpose, the following information is available via the Mode parameter:

- 0: Name of the customer to which the version is registered
- 1: Number of users in the ELO Client
- 2: Number of users in the Internet Gateway
- 3: Demo version (TRUE/FALSE)
- 4: Start version (TRUE/FALSE)
- 5: Full text license (TRUE/FALSE)
- 6: Replication license (TRUE/FALSE)
- 7: Backup server license (TRUE/FALSE)
- 8: Validity limit (date or empty)
- 9: Entire research IGW (TRUE/FALSE)
- 10: Dispatch folder (TRUE/FALSE)
- 11: Only e-mail filing (TRUE/FALSE)
- 20: Tobit mail interface (TRUE/FALSE)
- 21: COLD (TRUE/FALSE)
- 22: XML import (TRUE/FALSE)
- 23: SAPALINK interface (TRUE/FALSE)
- 24: http DocServer (TRUE/FALSE)
- 25: File reference generator (TRUE/FALSE)
- 26: Signature (TRUE/FALSE)
- 27: Batch scan (TRUE/FALSE)

All requests return a text with the number of users, user name (multiple lines) or a "TRUE" or "FALSE" for binary requests.

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

Parameter:

- Mode Select the serial number information 0..27

Return values:

- Value of the entry

Example:

```
' GetRegInfo.VBS  28.01.2004
'---------------------------------------------------------------
' © 2002 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo-digital.de)
'---------------------------------------------------------------
' This script reads the current registry information of the
' Access Manager. The GetRegInfo function expects the following
' parameters:
 0: Name of the customer to which the version is registered
' 1: Number of users in the ELO Client
' 2: Number of users in the Internet Gateway
```

```
' 3: Demo version (TRUE/FALSE)
' 4: Start version (TRUE/FALSE)
' 5: Full text license (TRUE/FALSE)
' 6: Replication license (TRUE/FALSE)
' 7:  Backup server license (TRUE/FALSE)
' 8: Date limit
' 9: ReadOnly IGW
' 10: Dispatch folder
' 11: Only e-mail repository
'12..19: Reserved
'20: Tobit interface
'21: COLD
'22: XML Importer
'23: SAPALINK interface
'24: HTTP Doc Server
'25: File number
'26: Signature
'27: Batch scan
' ------------------------------------------------------------

DIM Elo
DIM Text
DIM i
DIM Msg

Set Elo=CreateObject( "ELO.professional" )
Text=Array("User name","User number","Internet user",_
           "DemoVersion","StartVersion","Full text option",_
           "Replication","Backup server","Date limit",_
           "ResearchIGW","Dispatch folder","Only e-mail storage",_
           "Tobit interface",_
           "COLD", "XML Importer", "SAPALINK", "Doc server",_
           "File reference generator", "Signature", "Batch scan")

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
```

Available since: 3.00.282

1.133 GetScriptButton function

With this function, you can request the assignment of the script buttons within the ELO main screen.

The function is called within installation scripts, which allow scripts to be imported and buttons and menus to be assigned at the same time.

```
AnsiString GetScriptButton(int iTabSheet, int iButton)
```

Parameter:

- iTabSheet Select work area:

```
1: Repository work area (up to 16 buttons)
2: Clipboard (up to 8 buttons)
3: Intray (up to 8 buttons)
4: Search (up to 8 buttons)
5: Reminder (up to 8 buttons)
```

- iButton Number of the button (1..16 or 1..8)

Return values:

- The name of the script

See also:

- SetScriptMenu
- SetScriptButton
- ImportScript

1.134 GetScriptEvent (AnsiString Event, int Mode) function

This function reads the script name or complete path of the script set in the script events.

```
AnsiString GetScriptEvent (AnsiString Event, int Mode)
```

- Event: String with event identifier (see event list under SetScriptEvent)
- Or position when prefixed with a # (these values may vary in future ELO versions).
-
- Mode: 0 – Script name with complete path and file name extension
- 1 - Script name only

Return values:

- -2 - Unknown event
- -1 - Mode not implemented
- 0 - No script available

Available from: Version 3.00.228

Examples:

- Returns the complete path, with the filename of the script if available

```
Pfadkomplett = ELO. GetScriptEvent (“sEonTimer“,0)
```

- Returns only the script name, access via event identifier

```
NurName = ELO. GetScriptEvent (“sEonTimer“,1)
```

- Returns the script name, access via event position (#0 = "sEonTimer")

```
NurName = ELO. GetScriptEvent (“#0”,1)
```

See also:

- SetScriptEvent

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

This function returns the repository path for the current TreeView selection.

In version 6.0, the ObjectIDs list for the classic list view can be returned.

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

- Mode: 0: Short name, 1: ObjectID
- Delimiter: Separator between the individual entries
- MaxLength: Maximum output length. If this maximum length is exceeded, ELO removes a part.

Return values:

- -2 - Unknown event
- -1 - Mode not implemented
- 0 - No script available

Available since: 5.00.180

Example:

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

1.136 GetUILanguage function

With the help of this function, you can determine the current language setting of the ELO Client.

```
AnsiString GetUILanguage
```

Return values:

- "D": German
- "C": Czech
- "E": English
- "F": French
- "I": Italian
- "K": Slovak
- "N": Dutch
- "P": Polish
- "S": Spanish
- "DK": Danish

Available from: 6.00.054

1.137 GotoId function

Using this function, you can make ELO switch to a certain document (or folder). If you wish to link your own entries to ELO documents, it is enough to save the corresponding ELO ObjectID and call up the document via GotoID when you need to. You can use the GotoID(1) special case to return to the original Repository work area in one step.

Usually, the Repository work area is set up in such a way that the desired object is in the list on the left and is selected. The child entries or the image, if the object is a document, is displayed. Sometimes, it is useful to see the content of the object, not just the object itself. In this case, a minus sign should be added to the ObjectID.

```
int GotoId( int ObjektId )
```

Parameter:

- ObjectID ELO Object to be selected, or 0 (back to beginning of the repository)

Return values:

- -4: The user has no read permission
- -3: Path to the object not found
- -2: Object not found
- -1: No active work area
- 1: Ok

See also:

- LookupIndex
- GetEntryId
- GetEntryName

1.138 GotoPath function

Using this function, you can make ELO switch to a certain document (or folder). For this, you can enter the entire access path here and therefore can also describe paths via references in comparison to the GotoID command. The list with the ObjectIDs must be formed with the regular separators and must start with a "1".

Usually, the Repository work area is set up in such a way that the desired object is in the list on the left and is selected. The child entries or the image, if the object is a document, is displayed.

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

Parameter:

- ObjektIDs List of the ObjectIDs, each connected with a separator

Return values:

- -3: Path to the object not found
- -2: Invalid ObjectID in the path
- -1: No active work area
- 1: Ok

Available since: 6.00.000

Example:

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

See also:

- GotoId
- GetTreePath

1.139 Import function

Using this function, you can reimport an export data record.

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

Parameter:

- sSourcePath: Source path of the export record
- iParentID: Parent node of import object (1=repository)
- iSelII: ELO ObjectID of the import object

Return

- -1 No work area active
- -2 Error while importing
- 1 OK

Please note that the import writes a report file which you should use for evaluating possible errors in the Intray of the active user.

1.140 ImportScript function

With this function, a script (*.VBS file) can be imported to the ELO script directory.

The function is called within installation scripts, which allow scripts to be imported and buttons and menus to be assigned at the same time.

```
int ImportScript(AnsiString SourceFile, int ForceOverwrite)
```

Parameter:

- SourceFile Path + name of the script file (e.g. "A:\Script1.VBS")
- ForceOverwrite Force any already existing script to be overwritten Warning: Parameter results in different behaviors in ELOprofessional and ELOoffice. In ELOprofessional, a value of '0' causes an error; in ELOoffice a value of '1' causes an error. (Problem is known but will not be fixed.)

Return values:

- -3: Source file does not exist
- -2: Copying error
- -1: No active work area
- 1: Ok

See also:

- SetScriptMenu
- SetScriptButton
- GetScriptButton

1.141 InsertDocAttachment function

Using this function, you can add a file attachment to an existing ELO document. Remember that depending on the document options set, any data link, which already exists will be moved into the history list or overwritten. If the document has "version-proof" status, this function call will be refused.

```
int InsertDocAttachment( int ParentDoc, AnsiString DocumentFile )
```

Parameter:

- ParentDoc Internal ELO ObjectId of the document to which the file should be linked.
- DocumentFile Access path and name of the file that is to be linked.

Return values:

- -2: Operation stopped with repository or database error
- -1: No active work area
- 1: Ok

See also:

- UpdateDocument
- GetDocumentPath

1.142 InsertDocAttachmentEx function

Using this function, you can add a file attachment to an existing ELO document. Remember that depending on the document options set, any data link, which already exists will be moved into the history list or overwritten. If the document has "version-proof" status, this function call will be refused.

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

Parameter:

- ParentDoc Internal ELO ObjectId of the document to which the file should be linked.
- DocumentFile Access path and name of the file that is to be linked.
- Comment Comment (e.g. original file name) for the attachment
- Version Internal, freely definable version number or label

Return values:

- -2: Operation stopped with repository or database error
- -1: No active work area
- 1: Ok

Available from: 3.00.176

See also:

- UpdateDocument
- GetDocumentPath

1.143 InsertProjectOptions function

Using this function, you can add to the project options in the activities list.

For registering a new project, first create an entry under the pseudo-project "ELO_SYSTEM" with major number 1 (you can set the minor number to 0, the next free number will then be allocated automatically) and the entry value <projectname>. Under this project name, you can then save the lists for the sending method field, return status, recipients and 10 user-defined fields.

A list always consists of a heading with the minor number 1, and then an arbitrary number of options for the field value. If the heading starts with an exclamation mark, the user can only select an entry from the list. In all other cases the user is free to enter any value they wish.

If no options list has been defined for one of the 10 user-defined fields, the field will not be shown in the editing form. Even if you cannot make any specifications as a list of key words, you must at least enter a heading (minor number 1).

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

Parameter:

- Project Project name for which the lists are created.
- Major 10 Recipient list
- 11 Send Mode
- 12 Documentation status on return
- 30..39 User-defined field 1..10
- Minor Sequential number 1: heading, 2: option 1, 3: option 2...
- Value Display value

Return values:

- -2: No project name
- -1: No active work area
- 1: Ok

Example:

```
' First, the already existing entries are removed, and the
' script always enters a complete project data record
call Elo.DeleteProjectOptions( "ELO" )

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

' The recipients list (major no. 10) is defined. The
' user can only select one entry from this list, own entries are
' not possible
call Elo.InsertProjectOptions( "ELO", 10, 1, "!Recipient" )
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" )

' The sending method list is also a predefined field without
' any possibility of entering personal data
call Elo.InsertProjectOptions( "ELO", 11, 1, "!Recipient" )
call Elo.InsertProjectOptions( "ELO", 11, 2, "For viewing")
call Elo.InsertProjectOptions( "ELO", 11, 3, "For approval")
call Elo.InsertProjectOptions( "ELO", 11, 4, "For editing")

' Receipt status
call Elo.InsertProjectOptions( "ELO", 12, 1, "!Status" )
call Elo.InsertProjectOptions( “ELO”, 12, 2, “Approved” )
call Elo.InsertProjectOptions( "ELO", 12, 3, "Edited" )
call Elo.InsertProjectOptions( "ELO", 12, 4, "Unchanged" )

' First user-defined field, name "Product", selection from
' the list only
call Elo.InsertProjectOptions( "ELO", 30, 1, "!Product" )
call Elo.InsertProjectOptions( "ELO", 30, 2, "ELOprofessional")
call Elo.InsertProjectOptions( "ELO", 30, 3, "ELOoffice" )
call Elo.InsertProjectOptions( "ELO", 30, 4, "ELOviewer" )

' A further user-defined field, the user can select from the
' list or enter any value of choice
call Elo.InsertProjectOptions( "ELO", 33, 1, "Submodule" )
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" )

' Another field, not a default list
call Elo.InsertProjectOptions( "ELO", 34, 1, "Additional request" )
```

Available from: 3.00.360

See also:

- DeleteProjectOptions
- EditActivity
- ReadActivity
- WriteActivity
- NextActivity

1.144 InsertRef function

In addition to its hierarchical tree structure (cabinet - folder - tab - document), ELO also lets you create other references. A document called "Miller invoice" can be filed into the "Invoices" folder, while an additional reference is created in the "Miller" folder. Although there then is only one instance of that document in the system, it can be viewed and processed in both places.

The parameter OldParent determines whether you want to create a new reference (OldParent=-1) or move an existing reference (OldParent > 1).

If you are sure that your parameters are absolutely correct, you can switch type checking off using the CheckTypes parameter to increase efficiency. You are then responsible for the reference being executed correctly for the type, so a tab (3rd-level folder) may not be referenced out of a cabinet (top-level folder) or another tab, only out of a (2nd-level) folder.

If there is already a link from "NewParent" to "ObjID", this is not reported as an error. No double link is created either; the original situation remains.

You can also move an object with the help of this object (replace the OldParentID with an ObjectID).

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

Parameter:

- ObjID Internal ELO ObjectID of the entry referenced
- OldParent -1=new reference, otherwise an existing reference is moved
- NewParent Object ID of the referencing entry
- CheckTypes 0: No check; 1: Perform check

Return values:

- -10: Database errors when entering a reference
- -6: A cabinet cannot contain child entries
- -5: Error reading new parent node data
- -4: Error reading old parent node data
- -3: Error reading object data
- -2: Error moving the reference in the database
- -1: No active work area

See also:

- MoveToArchive
- LookupIndex

1.145 InsertVTRep (int, int, AnsiString) function

This function is used for creating your own text file for full text keywording.

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

Parameters

- ObjID: Document for which the full text file is to be created
- Flag: Bit 0: Sets flag "enter into the full text".
- Bit 1: Overwrite existing full text information
- FileName: Name of the text file containing the full text information (*.TXT).

Return values:

- -1: No active work area
- -2: Error reading the database information
- -3: Full text file already exists
- -4: Copying error
- -5: Error transferring the file to the Access Manager
- -6, -7: Target path could not be found
- >0: ok, file ID

Available from: 3.00.170

1.146 IntToDate function

This function converts an internal ELO date value into a date text.

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

Parameter:

- Date The date value to be converted.

Return values:

- Date text

Available from: 3.00.196

See also:

- DateToInt

1.147 LoadPostImg function

This function loads a file into the viewer in the Intray work area. The document does not need to be in the ELO Intray.

```
int LoadPostImg( AnsiString FileName, int Flags )
```

Parameter:

- FileName File to be loaded including path
- Flags Reserved, should be set to 0

Return values:

- -1 no active work area
- -2 Loading error
- 1 OK

1.148 LoadUserName function

This function finds the user name for a user ID. If the user does not exist or a wrong ID has been entered, an empty string is returned.

```
AnsiString LoadUserName( int UserId )
```

Parameter:

- UserID Internal ELO user number.

Return values:

- User name

See also:

- ActiveUserId
- FindUser
- SelectUser

1.149 LockObject function

If you want to edit an existing entry and need to make sure that it is not changed at the same time from somewhere else, you can lock it via this function at the beginning of your work and then release it again after you have finished. If a coworker wants to edit this objects during this time, a message is displayed that this entry is already blocked.

```
int LockObject( int ObjektId, int ActionCode )
```

Parameter:

- ObjektId Internal ELO object ID for the object to be locked or released
- ActionCode 0=approval for timestamp entry
- 1=lock with timestamp entry
- 3=query lock
- 8=release without a timestamp entry
- 9=lock without a timestamp entry

Return values:

- -4: (for ActionCode 3) the data set is not locked
- -3: invalid ActionCode
- -2: lock failed (is, for example, held by another user)
- -1: No active work area
- 1: (Action code 0 or 1) Ok
- 0..n: (ActionCode 3) User ID of the user who has locked the entry

1.150 Login function

If your program finds an ELO instance already running, it can work within the context of the user already logged in.

If your program, however, must start a separate ELO (such as a server process), it is necessary that you log in first. In the framework of this login, you transfer the desired sign-in names, the password and the repository to be edited.

After completion of the work, you must log out (log in with the user ID "LOGOUT"), otherwise the system will return a fault with a possible access conflict at the next start. You have two options for leaving the program, a normal exit (by exiting the program) through the "LOGOUT" ID and an incomplete exit returning to the login dialog using "LOGOUTNOQUIT" (e.g. to log in again under another name or work on another repository).

If you have carried out a complete logout, you must not attempt to log in again immediately. Since the program is in the process of exiting, you will encounter a segmentation violation.. You must wait for a short time (1..5 seconds) while the program ends completely AND then create a new ELO object with the CreateOleObject function (the old instance is no longer valid, its attempted use will lead to segmentation violation).

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

Parameter:

- UserName ELO user ID under which actions are to be executed
- Password ELO access password
- ArchiveName Repository required

Return values:

- -1: Login failed
- 1: Logout ok
- 0..255: Login ok, return of connection ID

1.151 LookupDelimiter (AnsiString) property

This function reads or sets the separating symbol for the LookupIndex function (and for related actions, such as the COLD column index). This change affects the entire ELO operation, not only the activities of the OLE Automation Interface.

Available from: 3.00.170

See also:

- LookupIndex

1.152 LookupDocType function

Detects the predefined ELO document type for a file name based on the extension. The file name can be available complete with a path, only as a file name or also only as an extension.

```
int LookupDocType( AnsiString FileName, int DefaultType )
```

Parameter:

- FileName File name with extension
- DefaultType Return value for the case that no assignment was found

Return values:

- Document type or default type

Available from: 4.00.034

Example:

```
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
```

1.153 LookupHistMD5Ext function

1.154 LookupHistMD5Ext function (obsolete)

This function looks up the number of documents that have a predefined MD5 hash value. This value is only looked up for newly created documents if the CheckSumIn switch in Access Manager is set to true. Via the mode parameter, you can determine whether deleted documents should be shown by setting the 0 bit position.

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

Parameter:

- MD5 The MD5 value to be looked up (a 32-byte hex character string).
- Mode 0: Show deleted documents
- 1: Omit deleted documents

Return values:

- -1 No active work area
- 0..n Number of documents with this hash value

Available from: 3.00.170 and 3.00.332 (ext. version)

See also:

- GetMD5Hash
- GetHistObj
- GetHistDoc

1.155 LookupIndex function

Finds the internal ELO Object ID from an access path. Submit an object index to this on the entry you are looking for, and the corresponding object ID will be returned. You can use this function, for example, to search for the object ID of a particular folder in which you want to file a new document.

The ObjectIndex can have various forms:

Entry in the chaos folder (only permitted for documents):

The property ObjIndex contains an empty entry (ObjIndex=""). No filing location is given, so the document will not appear in the filing structure, and can only be shown after a search.

Entry via an internal ELO object ID:

ObjIndex contains the object ID of a parent node (ObjIndex="#12345") prefixed by a # symbol. It is your responsibility to make sure that the parent node exists and has the right type.

Entry via an access path:

ObjIndex contains an access path to the parent node that starts with a ¶ symbol (ObjIndex="¶Cabinet¶Folder¶Tab"). This path is composed of the short names separated by the "¶" symbol (Alt-0182). Remember that using this approach, the same term should not be used twice on this level; otherwise, there can be no unique assignment. For example, there should not be two child folders for the month of March in the Invoices folder. On the other hand, the "March" folder can be used in any other folder.

Entry via a key word (documents only):

You can save up to three key terms in each folder (such as CSTNO 123). If you save the text CSTNO = 123 in ObjIndex, the folder named above will be used as the parent node.

Record via *Keytext (???)

```
int LookupIndex( AnsiString ObjektIndex )
```

Parameter:

- ObjIndex Access path to object required

Return values:

- ObjectId searched for
- -2: Error
- -1: No active work area

1.156 LookupKeyName function

Determines the internal ELO key ID via the key name. Please note that this information comes from a cache local to the client system and newly created keys on other workstations may only be available after a restart.

```
int LookupKeyName( AnsiString KeyName )
```

Parameter:

- KeyName Name of the key whose number is required.

Return values:

- >=0: Key ID
- -1: Key not found

See also:

- ObjKey
- LookupUserName

1.157 LookupMaskName function

Determines the internal ELO MaskId via the keywording form name. Please note that this information comes from a cache local to the client system and newly created forms on other workstations may only be available after a restart.

```
int LookupMaskName( AnsiString MaskName )
```

Parameter:

- MaskName Name of the form whose ID is to be looked up

Return values:

- >=0: Form ID
- -1: Form not found
- -2: No form name given
- -3: No active work area

See also:

- ReadObjMask
- WriteObjMask

1.158 LookupUserName function

Determines the internal ELO User or GroupId via the user or group name. Please note that this information comes from a cache local to the client system and newly created users on other workstations may only be available after a delay or a restart.

```
int LookupUserName( AnsiString KeyName )
```

Parameter:

- UserName Name of the required user/group

Return values:

- >=0: User/Group ID
- -1: Name not found

Available since: 4.00.054

Example:

```
Set Elo=CreateObject("ELO.professional")
Name=InputBox("Please enter a user or group name")
MsgBox Elo.LookupUserName(Name)
```

See also:

- LookupKeyName
- ReadUser

1.159 MaskFlags (int) property

The MaskFlags property contains information about the type of the form and the default settings for document flags for new documents of this type.

- Bit 0,1:
  - 00 No version control, the document can be changed at will.
  - 01 Filing with version control, old versions are retained.
  - 10 Filing as read-only, no subsequent editing/processing possible.
  - 11 Reserved
- Bit 3:
  - 0 Not to be used as a keywording form
  - 1 Can be used as a keywording form
- Bit 4:
  - 0 Not to be used as a search form
  - 1 Can be used as a search form
- Bit 6:
  - 0 No full text post-processing
  - 1 Log on for full text post-processing.

All other bits are reserved and are only used for internal purposes. They must be set to 0.

See also:

- ObjFlags

1.160 MaskKey (int) property

The MaskKey property determines the key of a form definition. The key number results from the "key management" table. In ELO "System settings -> Keys...".

See also:

- DocKey
- DocKind
- DocPath

1.161 MergeImages/MergeImagesEx function

With this function, you can copy one TIFF file into another TIFF file (e.g. in order to black out certain areas in the image). Select the source file, the target file, the pages and the transparent color as well as the position in the target document. Please note, that the transparent color has to be declared as a 24-bit RGB value, for entire black and white documents as well.

1 bit (black and white) and 24 bit (color images) can be used as a basis for the merge operation; the source and target have to have the same color resolution. 4, 8 and 16 bit intermediate formats are not supported.

```
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: Source file (e.g. a black rectangle)
- SFileTgt Target file in which the source file is to be copied
- IPageSrc Page in source file (important for multipage documents)
- IPageTgt Page of target file
- LTransp Transparent color, which becomes transparent in the source document
- X,Y Target coordinates
- Flags Bit 0: 1= opaque drawing, 2=transparent drawing

Return values:

- 1: Ok
- -2: Error copying image areas

Available since: 4.00.094

Example:

```
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
```

1.162 MergePostPages function

This function serves to file pages in the Intray into a cabinet. It corresponds to the "merge pages" function in the context menu. This function affects the pages selected in each case.

Return values:

- -2: No entries selected in the Intray
- -1: No active work area

See also:

- SelectPostBoxLine
- UnselectPostBoxLine
- PostBoxLineSelected
- SelectAllPostBoxLines
- UnselectAllPostBoxLines

1.163 MinDocLevel (int) property

Using this property, the filing level for documents can be set or read. In this way, you can file documents on any level depending on the setting.

Writing access on this property is only available from version 3.00.420. Remember that a change in this value only affects the local client and is not saved.

Values:

- 1 Cabinet (top-level folder)
- 2 Folder
- 3 Tab
- ...
- 253 Folder
- 254 Document

1.164 MovePostboxFile/MovePostboxFile2 function

Moves or copies a file from the user's Intray to the Intray of another user (parameter iUser). Mode=0 means moving, Mode=1 means copying. A bit with the value 2 determines whether or not a message box is displayed for error situations.

The regular MovePostbox command is not listed in the Intray report, only actions via MovePostbox2 are logged.

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

Parameter:

- sDataFile File name
- iUser User ID of the recipient
- iMode 0=move
- 1=copy
- sReportParam Additional parameter for the Intray report (short name).

Return values:

- 1: File successfully moved/copied
- -1: Error moving/copying file

Available from: (MovePostbox2) 3.00.276, (MovePostbox) 3.00.216

See also:

- ActivePostFile

1.165 MoveToArchive/MoveToArchiveEx function

Using this function, you can transfer the active Intray entry (created via AddPostboxFile) into a repository.

The object index can take various forms:

Entry in the chaos folder (only permitted for documents):

The property ObjIndex contains an empty entry (ObjIndex=""). No filing location is given, so the document will not appear in the filing structure, and can only be shown after a search.

Entry via an internal ELO object ID:

ObjIndex contains the object ID of a parent node (ObjIndex="#12345") prefixed by a # symbol. It is your responsibility to make sure that the parent node exists and has the right type.

Entry via an access path:

ObjIndex contains an access path to the parent node that starts with a ¶ symbol (ObjIndex="¶Cabinet¶Folder¶Tab"). This path is composed of the short names separated by the "¶" symbol (Alt-0182). Remember that using this approach, the same term should not be used twice on this level; otherwise, there can be no unique assignment. For example, there should not be two child folders for the month of March in the Invoices folder. On the other hand, the "March" folder can be used in any other folder.

Entry via a key word (documents only):

You can save up to three key terms in each folder (such as CSTNO 123). If you save the text CSTNO = 123 in ObjIndex, the folder named above will be used as the parent node.

```
int MoveToArchive( AnsiString ObjektIndex )
```

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

Parameter:

- ObjektIndex Filing destination in the filing system
- VersionNo Version number for the version history of the first document
- VersionComment Version Comment

Return values:

- -4: Could not move document into the repository.
- -3: Error transferring from the Intray to the repository
- -2: Unknown or missing filing destination
- -1: No active work area
- 1: Ok

See also:

- LookupIndex
- AddPostboxFile

1.166 MsgBox function

Displaying a message box.

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

Parameter:

- Correspond to those of the Microsoft default message box

Available since: 7.00.052

Example:

```
Call Elo.MsgBox ("Message", "Title", 0)
```

1.167 NoteOwner(int) property

This property contains the owner of the active "sticky" note in the script event call for editing a sticky note. The script event is activated before the editing dialog box and after the editing dialog box. Before the dialog box, the ScriptActionKey is set to 1, and after editing, it is set to 2 for Ok and 3 for Cancel.

Available since: 3.00.330; 5.00.224

See also:

- NoteText
- NoteType
- NoteAcl

1.168 NoteText (AnsiString) property

This property contains the text for the active sticky note for the "When editing a sticky note" script event request. The script event is activated before the editing dialog box and after the editing dialog box. Before the dialog box, the ScriptActionKey is set to 1, and after editing, it is set to 2 for Ok and 3 for Cancel.

Available since: 3.00.330; 5.00.224

Example:

```
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
```

See also:

- NoteType
- NoteOwner
- NoteAcl

1.169 NoteType(int) property

This property contains the type of the active "sticky" note (sticky note, personal sticky note, stamp) in the script event call for editing a sticky note. The script event is activated before the editing dialog box and after the editing dialog box. Before the dialog box, the ScriptActionKey is set to 1, and after editing, it is set to 2 for Ok and 3 for Cancel.

Available since: 3.00.330; 5.00.224

See also:

- NoteText
- NoteOwner

1.170 ObjAcl (AnsiString) property

With the ObjAcl property, you can either set or read the AccessControlList in the current entry. For a reading query, at least one read access is necessary for the object, and one write access for writing.

When you read this property, you will be given a text in the form <entry>, <entry>,…<entry>.

Under entry, first of all you will find the code for the type of access, and then the ID of the key, user or group involved. The indicator is always one character long; these are the possibilities:

- K This is a key entry
- R A user or group entry with read access rights
- W A user or group entry with write access rights
- D A group or user entry with delete access rights
- E A group or user entry with check-in access rights
- L A user or group entry with the "edit lists" right.

The indicators R, W, D, E, L can be combined, K must always be alone with a key number.

Example:

They are given the entry "K2,R3,RW4,RWDEL5". Key 2 is set; user or group 3 has read access rights, 4 has read/write access rights, and 5 has full access rights.

Available from: 4.00.000

Example:

```
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
```

See also:

- PromoteAcl

1.171 ObjBarcodeInfo (AnsiString) property

The ObjBarcodeInfo property can be used for querying the barcode configuration for the document type selected. The text is entered via the BarcodeInfo field in the keywording form manager and submitted for internal barcode analysis. However, you can also evaluate this text externally, for example if you wish to insert your own barcode components.

Example:

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

Available from: 3.00.264

1.172 OjbFlags (int) property

The ObjFlags property has various meanings depending on the type of object. If the object type is a folder, ObjFlags contains the sorting order of its child objects.

#define PLO_MANUAL 0 // Manual sorting order

#define PLO_ALPHA 1 // Alphabetic order

#define PLO_XDATE 2 // Document date (only useful for folders)

#define PLO_IDATE 3 // Filing or creation date

#define PLO_IXDATE 4 // Inverse – document date

#define PLO_IIDATE 5 // Inverse – filing date

#define PLO_IALPHA 6 // Inverse – alphabetic order

#define MFG_ISREPLROOT 0x80000 // Replication set start node

If this object is a document, the flags will contain the version control level and fulltext status.

- Bit 0,1: 00 No version control, the document can be changed at will.
- 01 Filing with version control, old versions are retained.
- 10 Genuine document filing, no subsequent editing/processing possible.
- 11 Reserved
- Bit 6: 0 No full text post-processing
- 1 Log on for full text post-processing.
- Bit 29: 1 The version history is searched during the search

All other bits are reserved and are only used for internal purposes. They must be set to 0.

Example:

Reading out the information:

```
If (Elo.ObjFlags And 1)= 1 Then ' Version control on
If (Elo.ObjFlags And 2)= 2 Then ' Stored revision-secure
If (Elo.ObjFlags And 64)= 64 Then ' Document in the full text database
If (Elo.ObjFlags And 256)=256 Then ' Document encrypted
```

Placing the information:

```
Elo.ObjFlags=Elo.ObjFlags Or 1 ' Version control on
Elo.ObjFlags=Elo.ObjFlags Or 2 ' Stored revision-secure
Elo.ObjFlags=Elo.ObjFlags Or 64 ' Document in the full text database
Elo.ObjFlags=Elo.ObjFlags Or 256 ' Document encrypted
```

Deleting the information:

```
Elo.ObjFlags=Elo.ObjFlags Or 1 ' Version control on
Elo.ObjFlags=Elo.ObjFlags XOr 2 ' Stored revision-secure
Elo.ObjFlags=Elo.ObjFlags XOr 64 ' Document in the full text database
```

```
Elo.ObjFlags=Elo.ObjFlags XOr 256 ' Document encrypted
```

See also:

- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- MaskFlags

1.173 ObjGuid (AnsiString) property

The ObjGuid contains the internal ELO global unique object identifier of the current entry. This GUID is unique globally for every ELO object and also stays the same for the replication. This property can only be read and is only available when the replication is turned on.

Maximum length: 32 characters

Available from: 3.00.180

See also:

- GetGuidFromObj
- GetObjFromGuid

1.174 ObjIDate (AnsiString) property

The ObjIDate property contains the filing date of the document or the filing structure element. The date must be entered in the format set in the system, e.g. DD-MM-YYYY or MM/DD/YYYY.

Remember that it makes no sense to set the property before filing to the repository (e.g. in the Intray or during filing), since this value will always be overwritten by the client software with the current date on filing. If you have to change the entry for whatever reason, you should do this after filing.

Maximum length: 12 characters

See also:

- ObjSDate
- ObjXDate

1.175 ObjIndex (AnsiString) property

The ObjIndex property contains the filing location of the current object. This filing location must be stored according to type. Therefore, you may only select a cabinet (top-level folder) as a filing location for a (2nd-level) folder and not a tab (3rd-level folder) or other folder.

The ObjectIndex can have various forms:

Entry in the chaos folder (only permitted for documents):

The property ObjIndex contains an empty entry (ObjIndex=""). No filing location is given, so the document will not appear in the filing structure, and can only be shown after a search.

Entry via an internal ELO object ID:

ObjIndex contains the object ID of a parent node (ObjIndex="#12345") prefixed by a # symbol. It is your responsibility to make sure that the parent node exists and has the right type.

Entry via an access path:

ObjIndex contains an access path to the parent node that starts with an exclamation mark. (ObjIndex="¶Folder1¶Folder2¶Folder3"). This path is composed of the short names separated by the "¶" symbol (Alt-0182). Remember that using this approach, the same term should not be used twice on this level; otherwise, there can be no unique assignment. For example, there should not be two child folders for the month of March in the Invoices folder. On the other hand, the "March" folder can be used in any other folder.

Entry via a key word (documents only):

You can save up to three key terms in each folder (such as CSTNO 123). If you save the text CSTNO = 123 in ObjIndex, the folder named above will be used as the parent node.

Record via *Keytext (???)

Maximum length: 200 characters

See also:

- LookupIndex
- ObjMName

1.176 ObjInfo (int) property

The ObjInfo property contains a freely definable value in interface programming. In an import process, the ObjId is stored from the old repository position, and in an automatic Office Professional repository comparison in the office version, the ObjId from the professional central repository will be stored here.

See also:

- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- ObjSReg

1.177 ObjKey (int) (invalid) property

The ObjKey property sets or reads the key of an entry.

See also:

- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- ObjSReg

1.178 ObjLock(int) read only property

The ObjLock property reads the lock on the current entry. The value -1 means that there is no lock on the entry; all other values return the UserId of the lock's owner.

Available from: 3.00.188

Example:

```
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 "No user lock"
  else
    MsgBox "Locked by " & Elo.LoadUserName(LockUser)
  end if
end if
```

See also:

- ObjShort
- ObjMemo
- ObjIDate
- ObjXDate
- ObjSReg

1.179 ObjMainParent (int) property

The ObjMainParent determines the main parent of a folder or document. Note: If this property is changed, you must make sure that the new main parent of a folder is always a cabinet, the main parent of a tab is a folder, and the main parent of a document is a tab. Errors can lead to undefined behavior in ELO (for example, the GotoObject function can no longer be carried out).

See also:

- DocKey
- DocKey
- DocKind
- DocPath

1.180 ObjMaskNo (int) property

The ObjMaskNo property sets or reads the document type currently set.

See also:

- ObjFlags

1.181 ObjMemo (AnsiString) property

Using this property, you can set the memo text for the current object. This memo text may contain a general description of the entry and is available for all object and document types.

Maximum length: 30000 characters (incl. ObjMemoInfo)

See also:

- ObjShort
- ObjFlags
- IDate
- Xdate
- ObjMemoInfo

1.182 ObjMemoInfo (AnsiString) property

Using this property, you can set invisible text in the memo field of the current object. This MemoInfo text can only be read or written via an automation interface and is available for all object and document types.

Maximum length: 30000 characters (incl. ObjMemo)

See also:

- ObjShort
- ObjFlags
- IDate
- Xdate
- ObjMemo

1.183 ObjMName (AnsiString) property

Reads/writes the short name of the currently active form (document type). This field is set when reading or creating an object (PrepareObject) from the form definition table. Remember that by changing this entry, the document type is not changed (this was set permanently by PrepareObject).

Maximum length: 40 characters

See also:

- ReadObjMask
- WriteObjMask

1.184 ObjOwner (int) property

With the ObjOwner property, you can look up or change the owner of an object.

See also:

- ObjInfo

1.185 ObjSDate (AnsiString), ObjSIDate, ObjSVDate property

The ObjSDate property contains a second document date for a search using a date range. The date must be entered in the format set in the system, e.g. DD-MM-YYYY or MM/DD/YYYY. Use ObjXDate to determine the start date, ObjSDate to set the end date of the search range. ObjSDate deals with the second object date, ObjSIDate the second filing date and ObjSVDate the second validity end date. These entries are only filled using search forms and are undefined in repository objects.

Maximum length: 12 characters

See also:

- ObjIDate
- ObjXDate

1.186 ObjSReg (AnsiString) property

The ObjSReg property contains the short name of a tab on the index tab. In all other objects, this value is accepted and saved, but is not shown or editable from within ELO.

Maximum length: 8 characters

See also:

- ObjInfo

1.187 ObjShort (AnsiString) property

Using the ObjShort property, you can read or write into the current object's short name. This text should (must) be entered for every object since this is the only means of orientation for the user in the filing structure view.

Maximum length: 50 characters

See also:

- ObjMemo
- ObjFlags

1.188 ObjStatus (int) property

You can determine via the ObjStatus property whether an object is deleted (Status <>0). Do not use this property to delete entries by simply writing a 1 on it.

See also:

- ObjInfo

1.189 ObjType (int) (invalid) property

1.190 ObjTypeEx (int) property

Using the ObjType property, you can find out the object type. Repositories with a four-level hierarchy work with ObjType, and repositories of more than 4 hierarchy levels work with ObjTypeEx.

`ObjType`

- The following values are available:
- 1: Folder (top-level)
- 2: Folder
- 3: Folder
- 4: Document

`ObjTypeEx`

- The following values are available:
- 1: Folder (top-level)
- 2: Folder
- 3:...
- .
- 253: Folder
- 254.... Document

Exercise extreme caution when changing the object type. Normally, there is no reason to change anything here. The object type is set when the entry is created according to the situation, and it therefore makes no sense to change it later. Do not attempt to use this function to create tabs within tabs, as this will unavoidably lead to inconsistencies in the database.

See also:

- ObjKey
- ObjFlags
- ObjKind

1.191 ObjVDate (AnsiString) property

The ObjVDate property contains the validity end date of a document. It must comply with the date format currently set in the system, i.e. usually in the form DD.MM.YYYY. If you want to run a search on the validity end date, enter the range in ObjVDate (from) and ObjSVDate (to).

Maximum length: 12 characters

See also:

- ObjSDate
- ObjXDate
- ObjIDate
- ObjSVDate

1.192 ObjXDate (AnsiString) property

The ObjXDate property contains the document date (for invoices, for example, the invoice date). The date must be entered in the format set in the system, e.g. DD-MM-YYYY or MM/DD/YYYY.

Maximum length: 12 characters

See also:

- ObjIDate
- ObjSDate

1.193 OcrAddRect function

Adds another rectangle into the OCR identification rectangle list. For the construction of a rectangle list it should be established a defined status with OcrClearRect, after that follows a sequence with at most 32 OcrAddRect commands. The rectangle is assigned in form of a string with the left upper coordinates and the right lower coordinates. All values are expressed in per thousand (e.g. 0,0,999,999 is the whole page).

```
int OcrAddRect( AnsiString rectangle )
```

Parameter:

- Rectangle in form of xa, ya, xe, ye

Return values:

- -1: Faulty rectangle list
- 1: Ok

1.194 OcrAnalyze and OcrAnalyzeEx function

The function OcrAnalyze passes on the name of the file to be evaluated and import with the default rectangle list the text into the text list.

```
int OcrAnalyze (AnsiString File name, int page number)
int OcrAnalyzeEx (AnsiString file name, int page number, int mode)
```

Parameters

- FileName: Name of the file to be analyzed
- PageNumber: Page to be analyzed (first page is 0)
- Mode: 1: Detect only digits
- 0: Detect all characters

Return values:

- -1: Error during OCR detection
- -2: OCR Engine could not be initialized
- 1: Ok

1.195 OcrClearRect function

Deletes the internal rectangle list of the OCR detection. This step should be accomplished as preparation before setting a new rectangle list, so that a defined status exists.

```
int OcrClearRect()
```

Parameter:

- None

Return values:

- always 1, ok

1.196 OcrGetPattern function

Returns a detected partial text of a pattern. Note, that each part of a pattern contains a partial text, even if it is actually permanent or empty. The pattern "*'invoice'_N*" has as first part of the pattern a * (any character string), after that follows as second part the permanent text 'invoice', the third part is a string of empty characters (which can also be empty), the fourth part is a number und at last as the fifth part follows any text again, the complete rest after the detected number can also be empty again. Please note that you have to request the first part with OcrGetPattern(0) (starting with 0, not with 1).

```
AnsiString OcrGetPattern ( int PatternNo )
```

Parameter:

- PatternNo: Number of the partial text out of the last pattern to be returned

Return values:

- Detected text (in case of an error an empty string is assigned)

1.197 OcrGetText function

After the OCR analysis the detected parts of the rectangle list are available in a text field. With the command OcrGetText you can readout these texts, OcrGetText(0) returns the text to the first rectangle, OcrGetText(1) to the second etc…

```
AnsiString OcrGetText( int TextNo )
```

Parameter:

- TextNo: Text to the n. entry in the rectangle list (first entry = 0)

Return values:

- Detected text (in case of an error an empty string is assigned)

1.198 OcrPattern function

The function assumes a pattern and an entered text and splits them according to the default pattern in partial texts. The structure of the pattern string you can gather from the appendix. The parameter PrepareText decides on the pre-processing of the incoming text before the pattern detection (to delete unnecessary white spaces (empty characters etc.)). Thereby the following combinations are available:

- 0: No WS will be deleted in the text
- 1: Several WS will be shortened to one
- 2: All WS will be deleted

Additionally you can add following values to this start value:

- 8: All WS will be deleted at the beginning and at the end of the text (approximates the TRIM() command in BASIC)
- 16: After every change of lines the WS will be deleted at the beginning of the line.

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

Parameter:

- PrepareText Source text pre-processing
- Pattern Pattern to be checked
- Text Text to be checked

Return values:

- -1: An error occurred while preparing the text
- -2: An error occurred while preparing the pattern
- -3: The pattern could not be found in the text
- >0: Pattern was found, the return value contains the amount of partial texts

1.199 OfficeMaskNo (AnsiString) property

The OfficeMaskNo property contains the configuration settings of the ELO Client under "System management" – "Options" – "General" – "Standard keywording forms for new entries" – "Microsoft Office documents". This property is read-only.

This property can only be read.

1.200 OkEnabled (int) property

With the OkEnabled property, you can check whether the OK button ies enabled in the ELO keywording dialog box and the user may make changes to the index data. The property must be checked, when the ELO keywording dialog box is replaced by a separate dialog box (e.g. a Visual Basic form). If the return value is 1, information is passed on to ELO that OK has been clicked by setting the property ScriptActionKey to 10. Therefore, it is is recommended to enable the OK button of the own form only when OkEnabled has the value 1.

This property can only be read.

Return values:

- 0: OK button not enabled
- 1: OK button enabled

1.201 (AnsiString) OpenSave (int Value) property

The values for the "Open" and "Save" dialog boxes are set or read using this property.

Value	Object	Input		R/W
1	DefaultExt	Text, 3 characters (the rest is truncated)		R/W
2	FileEditStyle	Value	Meaning	R/W
		Edit	Edit Box	
		ComboBox	Combo Box	
3	FileName	Text: <filename and path>		R/W
4	Files	Number of files selected with multiselect		R/-
5	Filter	Text: Text files (*.txt)|*.TXT| C++ files (*.cpp)|*.CPP		R/W
6	FilterIndex	Number: 1 to number of filter entries		R/W
7	InitialDir	Text: <Path with drive letter>		R/W
8	Options	See below (in connection with |)		R/W
9	Title	Text: <Title>		R/W
100…200	FileName	Selected file names where the AllowMultiSelect option is also set and several files have been selected.		R/-

Options:

value	Meaning
AllowMultiSelect	Allows users to select more than one file in the dialog box.
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 box) from the file selection dialog box. 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.

Available from: Version 3.00.228

Example:

Set the default extension to EXE

```
ELO. OpenSave (1) = „EXE“
```

Read the filename returned

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

See also:

- OpenSaveDialog

1.202 (AnsiString) OpenSaveDialog (int Type) function

This function creates an "Open" or "Save" dialog box depending on type. The values for title, FileName etc are accessible via the OpenSave property.

- Value Function
- 1 Open dialog box
- 2 Save dialog box

Return value:

- -1 - Not supported type
- 0 - Close dialog box with Cancel
- 1 – Dialog box closed via Ok button (OpenSave property is updated)

Example:

Creates an "Open" dialog box

```
ELO. OpenSaveDialog (1)
```

Creates a "Save" dialog box

```
ELO. OpenSaveDialog (2)
```

Values must be set beforehand with the OpenSave property. Otherwise, the dialog box appears with default Windows values.

Available from: Version 3.00.228

See also:

- OpenSave

1.203 OrientFile function

This function can be used for analyzing an image file in the Intray, any possible rotation through 90°, 180° or 270° is corrected. The analysis is performed using the OCR system. Before using the function, you need to run the OCRInit function, and the OCRExit function once it has ended. The function also works with multipage TIFF files.

As a file name, you can use the entry from the Intray (without the path, only the file name) or an index in the list of Intray documents (indicated by a # sign, e.g. #0 is the first entry in the list.)

```
int OrientFile( AnsiString FileName)
```

Parameter:

- FileName: Name of the file to be examined

Return values:

- -1: No active work area
- -2: No file name
- -3: Error loading the file
- -4: Error loading the page (OCR)
- -5: OCR error
- -6: Error saving the file
- 1: Ok

See also:

- GetDocumentOrientation
- RotateFile
- UpdatePostbox

1.204 OutlookName (AnsiString) property

This property returns the current name of the person or group to which the resubmission is to be sent.

See also:

- DelOutlookName

1.205 PopupObjID property

If using the "ClickOn" function with a function from the context menu, the object ID of the ELO entry to be edited must be assigned to the PopupObjID property.

Example:

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

1.206 PostBoxLineSelected function

This function can be used for testing whether a line in the Intray has been selected.

```
int PostBoxLineSelected( int LineNo)
```

Parameter:

- LineNo The line to be tested

Return values:

- -2: Invalid row number
- -1: No active work area
- 0: Line not selected
- 1: Line selected

See also:

- SelectPostBoxLine
- UnselectPostBoxLine
- SelectAllPostBoxLines
- UnselectAllPostBoxLines

1.207 PrepareObject (invalid) function

1.208 PrepareObjectEx function

This function reads an ELO object (folder or document) into an internal editing buffer. From here, various fields such as short name, color, or memo text can be read, processed and edited. These changes can then be transferred to the database permanently using the UpdateObject function.

If you wish to create a new object, you first have to initialize a new, empty entry with the ObjectId set to 0 using this function. If you wish to mark an existing entry as "new" (that means all data is retained, but a new entry is created when saved), enter -2 as the ObjectId. This function is available from Version 2.02.059 onwards.

The third possibility might be setting an Intray entry as an "active Intray entry". In this case, set ObjectId to -1 and the Intray line number (starting with 0) as the ObjectType.

If a search is to be performed after the PrepareObjectEx via DoSearch or DoSearchEx, transfer the value -3 as the ObjectId (starting with client version 6.00.054).

Repositories with a four-level hierarchy work with PrepareObject; repositories with more than four hierarchy levels work with PrepareObjectEx.

Remember that when reading Intray entries, the error conditions -5 and -7 are returned. These are only errors in the sense that "the key words could not be read since they have not been created". The Intray entry still exists without keywords and can be processed.

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

Parameter:

- ObjectId 0: for a new entry
- -1: for an Intray entry
- -2: current entry is marked "New"

Else: internal ELO object access ID to the object.

- ObjectType 1=folder, 2=folder, 3=folder, 4=document, if Intray entry line number (0..n)
- MaskNo Document type (see also ReadObjMask)

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

Parameter:

- ObjectId 0: for a new entry
- -1: for an Intray entry
- -2: current entry is marked "New"
- -3: Initialize the entry for the following search
- else. Internal ELO object access number to the object
- ObjectType 1=Folder, 2=Folder, 3=..., ..., 253=Folder, 254=Document, if Intray entry line number (0..n)
- MaskNo Document type (see also ReadObjMask)

Return values:

- -1: Unknown form type
- -2: Error reading the entry
- -3: No active work area
- -4: No Intray path found
- -5: The Intray entry does not yet have any keywords and is not selected
- -6: The Intray entry could not be found
- -7: The Intray entry does not have any keywords and is selected
- -8: No read access rights
- 1: New empty entry available
- 2: Existing entry read from the database
- 3: Existing, selected Intray entry read
- 4: Existing, non-selected Intray entry read
- 5: Existing, deleted but not yet permanently removed Intray entry read

Example:

Call the first Intray document, set the keywording form ID to 2 and automatically fill the short name and first index field. After that, the document is transferred to the repository using the path provided:

```
' Destination folder for the document
RegisterId = "¶Schrank¶Ordner¶Register"
MaskNo = 2
Set Elo=CreateObject("ELO.professional")

' First, update the Intray to be safe
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 "No document found in the Intray"
end if
```

See also:

- UpdateObject
- LookupIndex

1.209 (int) PrintDocList function

PrintDocList (AnsiString DocTitle, AnsiString ObjIdList, AnsiString ObjList, int Type)

This function prints an overview of the objects given.

If Type 1 is set, the column widths can be set via the PrintDocListTabs property. The value are set to DIN A4 portrait by default. Columns with a value of 0 are not printed. If the texts should go beyond the width, it is automatically shortened and three dots are added at the end.

- DocTitle: Heading (if "" is returned, the title is: search result)
-
- ObjIdList: List with object IDs separated by a separator. Unavailable object IDs are ignored.
- ObjList: List separated by a separator. Wrong entries are ignored.
- Short - Short name
- Date - Filing date
- Ddate - Document date
- Index - Index lines
- Memo - Memo text
- Type: 32 bit integer form
-
- 0 – List view
- 1 – Column view
- 0 – Short headings
- 1 – Long headings
- 0 – No separator line
- 1 – Separator line after each entry
- 0 – Do not show a dialog box
- 1 – Show a dialog box (values returned are used)
- x Currently unused (reserved for extensions)

Return value:

- -3 Dialog box ended via Cancel button
- -2 No objects available for printing
- -1 No active work area
- 0 Document sent to printer (OK button pressed in dialog box)

Example:

Prints a list of the ObjectIds provided in the short name, filing date and index fields. The heading is the search result.

```
ELO. PrintDocList (“”,”10¶11¶52”,“Short¶Date¶Index“,0)
```

Prints a table of the ObjectIds provided with the short term, filing date and index fields. The heading is "My search".

```
ELO. PrintDocList (“My search”, ”10¶11¶52”, “Short¶Date¶Index“,1)
```

Available from: Version 3.00.228

See also:

- PrintDocListTabs

1.210 PrintDocListTabs (int No) property

The tabs for the PrintDocList printout column view can be set using this property.

```
int PrintDocListTabs (int Nr)
```

- No.:
  - 0 Default
  - 1 Short name column
  - 2 Document date column
  - 3 Filing date column
  - 4 Index column
  - 5 Memo column

The column widths are to be given in mm.

The field with the number 0 is for entering preset values.

The following entries are permitted:

- 1 Sets the portrait default values
- 2 Sets landscape default values

Available from: Version 3.00.228

Example:

Sets the default values for portrait printing.

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

See also:

- PrintDocList

1.211 PrintDocument function

This function prints the currently displayed document.

```
PrintDocument(int MitDialog, String DeviceName, String Port, int FromPage, int ToPage, int Copies)
```

Parameter:

- MitDialog 1: Before printing, the printer selection dialog box appears
- 0: The printer selection dialog box is not shown
- DeviceName Name of the printer, can also be a network printer, if DeviceName="", the default printer is used
- Port Name of the printer port (only relevant if one and the same printer driver operates several printers through separate interfaces)
- if Port="", the default value is used
- FromPage,
- ToPage Document is printed page to page; if both values are "0", the entire document is printed
- Copies Number of copies

Return values:

- 1: Ok
- -1: ELO is not in the main work area
- -2: No document is shown
- -3: Printing error
-

1.212 PromoteAcl function

This function can be used to alter the child entries of an object to the current ACL. The organization of the ACL entries is described at the ObjAcl property.

```
int PromoteAcl ( int iObjId, int iShowResult, int iExact, int UpdateVal,
AnsiString mAcl, AnsiString mAdd, AnsiString mDel )
```

Parameter:

- iObjId Start node of the entries to be adjusted
- iShowResult 0: do not show result dialog box, 1: show a result dialog box
- iExact 0: mAdd and mDel ACLs contain the changes in the old ACL
- 1: mAcl contains the new ACL
- UpdateVal Levels to be noticed: Bit 0=structure elements, Bit 1=documents
- mAcl New access control list (ACL), used for all child entries
- where iExcat=1
- mAdd Additional entries for the ACL, used where iExcat=0
- mDel Entries to be removed from the ACL, used where iExcat=0

Return values:

- -1: No active work area
- 1: Ended

Available from: 3.00.204

See also:

- ObjAcl

1.213 QueryOption function

The QueryOption function can be used to query individual settings in the options dialog box.

```
AnsiString QueryOption( int OptionNo )
```

Parameter:

- OptionNo: Function list; see SetOption

Return values:

- Current value of the option or "ERROR"

Example

```
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
```

Available since: 3.00.286

See also:

- SetOption

1.214 ReadBarcodes function

This function reads barcodes located in an Intray TIFF document. The name of the Intray file can be transferred with SourceFile. Alternatively, a line number (#0, #1, #2, etc.) can be returned. The corresponding file in the Intray list is then used as the file name. The function returns the numer of recognized barcodes. The barcodes can be called up with the help of the *GetBarcode* function.

The BarcodeDescriptor parameter contains a list of the rectangles to be analyzed. Refer to the barcode documentation for a detailed description of this list.

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

Parameter:

- SourceFile File to be checked or mailing list line numbers (with #number).
- BarcodeDescriptor List of rectangles, in barcode format.
- MaskNo Keywording form number of the keyword file to be created
- ReadMultiple 1 = Read all barcodes in the rectangle defined
- 0 = Only read one barcode

Return values:

- >0: Number of recognized barcodes
- -1: No open workspace
- -2: No rectangle list available
- -3: Faulty rectangle list
- -4: File name does not exist
- -5: Barcode component could not be initialized
- -6: Invalid TIFF format

See also:

- GetBarcode
- AnalyzeFile

1.215 ReadColorInfo function

With this function, you can read a color definition of an ELO color object. You can access the color settings via the ColorInfo and ColorName properties.

If you need a list of the colors available, you can call the function starting with 0 and the bit set at 0x8000. The next corresponding color value is then loaded.

```
int ReadColorInfo( int ColorNo )
```

Parameter:

- ColorNo: Number of the color definition to be read

Return values:

- -1: No active work area
- -2: Invalid value for color number
- 1: Ok

See also:

- WriteColorNo
- ColorInfo
- ColorName

1.216 ReadKey (int) function

This function reads the system key entry and returns a name.

```
AnsiString ReadKey (int KeyNo)
```

- KeyNo: Key number starting with 1

Return value:

- Name of the key or empty string if the key number is invalid.

Example:

To return the name of the key with the number 1.

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

See also:

- WriteKey

1.217 ReadObjMask function

With this function, you can read a form (document type) into the internal ELO object. The attribute name and index fields, but not the entry value fields with the defined values, are preset.

```
int ReadObjMask( int MaskNo )
```

Parameter:

- MaskNo: Number of the document type form to be read in

Return values:

- -3: No active work area
- -2: Invalid value for form number
- -1: Error reading database
- 1: Ok

See also:

- WriteObjMask
- GetObjMaskNo

1.218 ReadSwl function

This function reads a layer from the branch of a keyword list. Every entry of a keyword list is given a unique indicator within its level – a 2-digit letter combination: AA, AB, AC up to ZZ. By concatenating all flags at all levels, you can specifically address every keyword in the tree. Starting with the point for the root, you can now, for example, delete the first entry (AA) below the second (AB) and below that the third (AC) entry using "AAABAC". For this, all entries below the selected level are returned together with the corresponding identifiers, separated by the separator are returned.

The group designates the assignment of a list to an index field (i.e. the name of the keyword list must be the same as the group field in the keywording forms editor).

```
AnsiString ReadSwl( AnsiString Gruppe, AnsiString Parent, AnsiString Delimiter)
```

Parameter:

- Group Selects a keyword list
- Parent Path to the entries to be deleted
- Delimiter Separator

Return values:

- -1: No open workspace
- -2: Error saving the keyword
- else: List of keywords with the respective identification

Available from: 5.00.066

Example

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

1.219 ReadUser function

The ReadUser function reads in a user data record from the system file. Administrators can read and edit any users, subadministrators can only read and edit some users. Normal users, however, may also read out some files of a user data set.

You can generate an empty user data set via UserNo -1. Via UserNo -2, you read all of the rights of the current users (that is the UserGroups property not only contains direct groups but also child groups, the same applies to UserKeys) (available starting Version 3.00.546).

If you want to set up a new user, then you must first look for a free user ID (a free user is a user without an ID). Then, you can fill in the user properties and save the data set (absolutely make sure that the user ID is correct as you could otherwise overwrite an existing user).

```
int ReadUser( int UserNo )
```

Parameter:

- UserNo: Number of the user to be read

Return values:

- -1 Error while reading the user from the Access Manager
- -2 Only admins and subadmins can read user data
- -3 A subadmin has attempted to read a foreign user
- 1 An empty user record has been created via UserNo=-1 or -2
- 2 The user record has been read correctly
- 3 The registered user is not the administrator of the read user and the UserFlags were set to 0.

See also:

- ReadUser
- UserGroups
- UserParent
- UserKeys
- UserFlags

1.220 ReadUserProperty function

The ReadUserProperty function reads a user extension from one of 8 user data fields. Remember that the first four fields are reserved by ELO (e.g. for NT name conversion). Fields 5 to 8 are freely available.

```
AnsiString ReadUserProperty( int PropertyNo )
```

Parameter:

- PropertyNo: Number of the property to be read

Return values:

- Property read or empty string for error

Example:

```
' AutoStart.VBS  20.03.2002
'-----------------------------------------------------------------
' © 2002 ELO Digital Office GmbH
' Autor: M.Thiele (m.thiele@elo-digital.de)
'-----------------------------------------------------------------
' This script reads a start folder or child folder from
' the user data and links to the specified location. Here,
' a separate path is specified for each repository. The definition occurs
' in the first user-defined field in the user manager
' in the format
' |<archivename1>¶Folder¶Folder|<archivename2>¶<anotherfolder>
' The individual repository definitions are also separated using a pipe(|)
' symbol.  Each definition starts with the
' repository name, followed by a repository path.
' ---------------------------------------------------------------

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
```

See also:

- WriteUserProperty
- UserGroups
- UserParent
- UserKeys
- UserFlags

1.221 ReadWv function

Reads a reminder (determined by the WvIdent parameter) into the internal reminder memory. This date can then be read out with the various property functions. If WvIdent has 0 value, the internal reminder memory is deleted; this operation is essential before creating a new date.

```
int ReadWv( int WvId )
```

Parameter:

- WvId: number of date to read, 0: create new, blank date

Return values:

- -1: No active work area
- -2: Error while reading
- 1: Ok

See also:

- WriteWv
- DeleteWv
- WvIdent
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.222 ReloadWv function

This function updates the list of reminders in the main ELO window.

```
int ReadWv( )
```

Parameter:

- None

Return values:

- -1: No active work area
- 1: Ok

See also:

- WriteWv
- DeleteWv
- WvIdent
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.223 RemoveDocs (int, AnsiString, int) function

Via this OLE function, you can automate calling up the dialog box for deleting legacy documents. Parameters submitted are the document path restriction, the cut-off date in ISO format (YYYYMMDD) and a mode parameter with the coded actions.

The mode parameter is composed of the following values:

Either 1,2 or 3 for the document control. For 1, the document is deleted without an additional check (very dangerous, usually do not use). For 2, a size comparison is performed and for 3, the file content is compared (analogous to the options from the dialog box.).

With the 1,2 or 3 mode, only a deletion check is performed initially. No files are actually deleted at this point, but there is only a report on how many documents were deleted. Deletion only takes place once you add the number 21840 to this value - therefore 21841, 21842 or 21843.

As a return value you get an error number or a deletion report. The deletion report contains a series of numbers separated by a comma.

- 1 Last document number
- 2 Number of deleted files
- 3 Number of files without a backup
- 4 Number of defective entries
- 5 Size of the deleted documents (false output if more than 2GB).

```
AnsiString RemoveDoc (int PathId, AnsiString cut-off date, int Mode)
```

- PathId : Path to be deleted (0 = all paths)
- Cut-off date : Cut-off date in ISO format, only older entries are deleted
- Mode: 1,2 or 3 for deletion control, 21841, 21842 and 21843 for deleting documents

Return value:

- -1,… - Error while deleting
- -2 - Invalid cut-off date
- -3 - No active work area
- Else -Deletion report

Available since: 5.00.094

Example:

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

1.224 RemoveRef (int, int) function

In addition to its hierarchical tree structure (cabinet - folder - tab - document), ELO also lets you create other references. A document called "Miller invoice" can be filed into the "Invoices" folder, while an additional reference is created in the "Miller" folder. Although there then is only one instance of that document in the system, it can be viewed and processed in both places.

The function also lets you remove a reference already created.

```
int RemoveRef (int ObjId, int RefNo)
```

- ObjId : Internal ELO ID
- RefNo : Number of reference to be deleted (starting with 1)

Return value:

- 0 Reference deleted
- -1 - No active work area
- -2 - Reference not available
- -3 - Delete failed
- -4 - No read/write access to object
- -5 - Only additional references can be deleted

Example:

- Removes the fifth reference from object with ID 34.
- Elo.RemoveRef (34,5)

See also:

- InsertRef
- GetObjRef

1.225 RotateFile function

This function lets you rotate a graphics file in the Intray by 90, 180, or 270 degrees. In the current version, only single-page TIFFS can be rotated; multipage documents cannot be edited.

The file name given may be an item from the Intray (no path, just the file name), an index number in the mailing list (marked with #, e.g. #0 is the first item in the mailing list) or an empty string. In this case the current Intray file (ActivePostFile, e.g. from the previous scan) is used.

```
int RotateFile( AnsiString FileName,  int Degree )
```

Parameter:

- FileName: Name of file to be rotated
- Degree: Angle of rotation, 90, 180 or 270

Return values:

- -1: No active work area
- -2: No file name
- -3: Error reading file
- -4: File could not be rotated or saved
- 1: Ok

See also:

- AnalyzePostfile
- UpdatePostbox

1.226 RunEloScript function

This function runs an ELO script. The system searches for the script file in the *Intray\EloScripts* directory.

```
int RunEloScript(AnsiString FileName)
```

Parameter:

- FileName Script to be run (without extension)

Return values:

- -1: Script file does not exist
- 0: Ok

See also:

- DoExecute

1.227 SaveDocumentPage function

This function lets you save one page of a multipage TIFF file.

```
int SaveDocumentPage(AnsiString sFileName, int  iPage, int  iQuality)
```

Parameter:

- sFileName: Name of file to be saved
- iPage: Page number (indexed with "1")
- iQuality: Quality of JPEG graphics (0..100)

Return values:

- -1: No active work area
- -2: No viewer visible
- -3: Error saving
- 1: Ok

1.228 SaveDocumentZoomed function

This function lets you save the document currently displayed scaled down in proportion.

```
int SaveDocumentZoomed(AnsiString sFileName, int  iZoom, int  iQuality)
```

Parameter:

- sFileName: Name of file to be saved
- iZoom: Zoom factor in [%]
- iQuality: Quality of JPEG graphics (0..100)

Return values:

- -1: No active work area
- -2: No viewer visible
- -3: Error saving
- 1: Ok

1.229 SaveObject function

This function lets you save and restore the internal object keywording data. The function may not be used recursively, because there is only one backing store. Every Save action overwrites the previous one. The Restore function may be used repeatedly, but in that case the same data is restored each time.

```
int SaveObject( int Mode )
```

Parameter:

- Mode: 1: Save, 2: Backup

Return values:

- -1: Invalid parameter
- 1: Ok

1.230 ScriptActionKey (int) property

This property is used to return information to ELO from the script. ELO responds to it with very specific actions.

Program context	Value	Action by ELO
Calling a script when entering a keywording form	10	Leave keywording form with "OK"
	-10	Leave keywording form with "Cancel"
Scripts: before importing/exporting	0	Do not import/export
	1	Import/export
Controlling input focus in the keywording form in scripts running when you enter or leave input fields	11	"Short description" field gets input focus
	12	"Memo" field gets input focus
	13	"Date" field gets input focus
	1000+n (n=0-49)	Input line n gets input focus
Available from: Version 3.00.228		
After OK click in keywording form	-20	Cancel OK action
Available from: Version 3.00.350		
Sticky note	0	After editing a sticky note
	1	Before editing a sticky note
When creating/moving a shortcut to an object reference	-30	Moving is prevented with this

1.231 SearchListLineId function

With this function the ELO ObjectID of a line from the search view can be detected.

```
int SearchListLineId(int LineNo)
```

Parameter:

- LineNo Line to be selected

Return values:

- -2: Invalid row number
- -1: No active work area
- >0: Object ID

Available since: 5.00.036

Example:

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

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

MsgBox res
```

See also:

- UnselectSearchListLine
- SelectSearchListLine
- SearchListLineSelected

1.232 SearchListLineSelected function

This function lets you test whether a line is selected in the Search work area.

```
int SearchListLineSelect ed(int LineNo)
```

Parameter:

- LineNo Line to be selected

Return values:

- -2: Invalid row number
- -1: No active work area
- 0: Line not selected
- 1: Line selected

See also:

- UnselectSearchListLine
- SelectSearchListLine

1.233 SelectAllPostBoxLines function

This function selects all the lines of the Intray.

```
int SelectAllPostBoxLines()
```

Parameter:

- None

Return values:

- -1: No active work area
- 1: Ok

See also:

- SelectPostBoxLine
- UnselectPostBoxLine
- PostBoxLineSelected
- UnselectAllPostBoxLines

1.234 SelectArcListLine function

This function selects an entry in the Search work area. The line number here runs from 0 to the amount of entries minus 1.

```
int SelectArcListLine(int LineNo)
```

Parameter:

- LineNo Line to be selected

Return values:

- -2: Invalid line number
- -1: No active work area
- 1: Ok

Example:

```
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
```

See also:

- UnselectArcListLine
- ArcListLineSelected

1.235 SelectLine function

The SelectLine function lets you select an entry in the current selection list. To do this, the left-hand area of the Repository work area is used, or the list in the Clipboard, Intray, Search, or Tasks work area.

```
int SelectLine( int LineNo )
```

Parameter:

- LineNo: Line to be selected

Return values:

- -3: No active work area
- -2: No selection list available
- -4: Reaches end of list
- Else: Line selected before operation

See also:

- SelectView

1.236 SelectPostBoxLine/SelectPostBoxLineEx function

This function selects an entry in the Intray. By using SelectPostBoxLineEx you can load the document in the viewer again.

```
int SelectPostBoxLine(int LineNo)
int SelectPostBoxLineEx(int LineNo, int Mode)
```

Parameter:

- LineNo Line to be selected
- Mode 1=Document is reloaded in the viewer

Return values:

- -2: Invalid row number
- -1: No active work area
- 1: Ok

See also:

- UnselectPostBoxLine
- PostBoxLineSelected
- SelectAllPostBoxLines
- UnselectAllPostBoxLines

1.237 SelectSearchListLine function

This function selects an entry in the Search work area. The line number here runs from 0 to the amount of entries minus 1.

From version 3.00.544 on, you have the further possibility to enter a negative line number. In this case, the line is selected, and the document view for this line is also refreshed. Since for the first line it can't be distinguished between a "0" and a "minus 0", this line begins at "-1" instead of a "-0" for the first entry.

```
int SelectSearchListLine(int LineNo)
```

Parameter:

- LineNo Line to be selected

Return values:

- -2: Invalid row number
- -1: No active work area
- 1: Ok

See also:

- UnselectSearchListLine
- SearchListLineSelected

1.238 SelectTreePath function

This command activates the tree view in the Search work area and opens it up according to the specified parameter string. The various levels are each marked by a ¶-separator, the last level will be selected and not separated further. You can specify the placeholders "*" for "select first entry" and "-" for "select nothing" for the last level.

```
int SelectTreePath( AnsiString TreePath )
```

Parameter:

- TreePath Path to be separated in the tree view.

Return values:

- -1: No active work area
- 1: Ok

Available from: 4.00.152

Example:

The following example performs a search over all e-mails (form number 2), which contain any instance of "Thiele". Afterwards, the path warehouse – purchase – arrivals is separated and there the first child entry is selected:

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

Elo.SelectView 4
Elo.PrepareObjectEx 0, 0, 2
Elo.SetObjAttrib 0, "%Thiele%"
Elo.DoSearch
Elo.SelectTreePath "warehouse¶purchase¶arrivals¶*"
```

See also:

- SelectView
- SelectTreePathEx

1.239 SelectTreePathEx function

This command activates the tree view in the Search work area with a virtual view and opens it up according to the first parameter string. (See description of SelectTreePath)

Virtual view if defined via the 2nd parameter. If you enter a "*" here, a classic folder tree is displayed in this case.

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

Parameter:

- TreePath Path to be separated in the tree view.
- DefAnsicht Definition for a virtual view

Return values:

- -1: No active work area
- 1: Ok

Available from: 7.00.032

Example:

E-mails will be structured according to sender and recipient:

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

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

The text with the definition begins with the name: "MailFromTo". This name is followed by the pipe character as separator, the keywording form's number ("2"), and a colon as separator with an indicator for whether empty index entries should be filled (0 or 1, "1" for the example). The numbers of the index fields follow, separated by commas. There must always be exactly 5 entries. "-1" is used to indicate unused index fields.

See also:

- SelectView
- SelectTreePath

1.240 SelectUser/SelectUserEx function

The SelectUser and SelectUserEx function lets you call a selection dialog box to select a user. This returns a list of all users (optionally suppressing your own entry) and the user ID selected (or –1 if you cancel).

```
int SelectUser( int SuppressOwnName )
```

Parameter:

- SuppressOwnName 0: All users, including your own name
- 1: Your own name is not included for selection.

Return values:

- -1: No user selected
- >=0: UserId
-

```
AnsiString SelectUserEx( int SelectFlags )
```

Parameter:

- SelectFlags Value: 1: Allow MultiSelect in user selection
- 16: User names – use last name first
- 32: Group name – use last name first
- 256: Do not include own name in list

Return values:

- Empty: No user selected
- Else: user list, numbers separated by commas for multiple selection

Available: (Ex) 3.00.276

Example:

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

See also:

- FindUser
- LoadUserName
- ActiveUserId

1.241 SelectView function

The SelectView function lets you decide which working view you want to have on top. You can also use it to check which view is currently active.

```
int SelectView( int ViewId )
```

Parameter:

- ViewId0: Query current work area, return value 1..5
  - 1: Repository work area
  - 2: Clipboard
  - 3: Intray
  - 4: Search
  - 5: Reminder
  - 6: Messages
  - 7: Records (Government version only)
  - 8: In use

Return values:

- -2: Invalid value for ViewId
- -1: No active work area
- 1: When ViewId>0: new work area activated

See also:

- SelectLine

1.242 SelectWorkArea function

Normally all automation commands affecting the user interface (such as GotoId) apply to the currently active work area. If you are working with a second view, you cannot control the two windows independently of each other. The SelectWorkArea function lets you define all subsequent commands for one of the two views. After completing the window-based actions, you should reset to the default setting with SelectWorkArea(0). Watch out that, where operations extend over a period of time, a work area may have become invalid, e.g. if the user has closed the window in the meantime. For this reason the explicit selection should always only be made at short notice and should be reset immediately after completing the tasks.

```
int SelectWorkArea( int ViewNr )
```

Parameter:

- ViewNo. Number of the work area
- 0: uses area currently active (default setting)
- 1: uses area 1
- 2: uses area 2

Return values:

- -1: Invalid work area selected
- -2: Area selected not active
- 0,1,2: Area selected last

Available from: 3.00.264

Example:

```
set Elo = CreateObject("ELO.professional")
if Elo.SelectWorkArea(1)>=0 then
  ‘ here are the commands for the first work area
  x=Elo.SelectView(3) ’ Switch to Intray view
end if

if Elo.SelectWorkArea(2)>=0 then
  ‘ here are the commands for the second work area
  x=Elo.SelectView(5) ‘Switch to Reminder view
end if

'Reset to default setting after the action
x=Elo.SelectWorkArea(0)
```

1.243 SelList function

The SelList function lets you make ELO show a hierarchical list. Provide the file name of the list as an entry parameter. If you only provide a name without a file path, the file is expected in the Intray directory. This returns an empty string (Cancel) or the option selected.

```
AnsiString SelList( AnsiString )
```

Return values:

- Empty: Cancel, no selection
- Text: Option selected

1.244 SeparateTiffFile function

With the function SeperateTiffFile, you can split multipage tiff files into single tiff files. These files can then be automatically combined back to partial documents via separator pages.

The original name is used as the file name for the individual files. An underscore and a line number are added to the file name. Therefore, the file XYZ.TIF becomes the individual files XYZ_0.TIF, XYZ_1.TIF, which are stored in the same directory.

Caution: ELO does not check whether there is a name conflict for the newly created files. If there is an entry among one of the automatically generated file names, it will be overwritten without a warning. If it is not guaranteed that such conflicts cannot occur, the script needs to perform this check itself before being called up.

```
AnsiString SeparateTiffFile( AnsiString FileName, int ScanProfile, int
separator, int empty page )
```

Parameter:

- FileName: Path and file name of the file to be split
- ScanProfil: -1: no default, 0..7: ELO scan profile (definition of the separator page, empty page)
- Separator page: Separator page to be used for partial document detection:
- 1: No separator page, all pages are put into one target document
- 2: Bar separator page
- 3: Blank page as separator
- 4: Single pages, no partial document compendium
- Empty page: 0: Maintain empty pages, 1: Discard empty pages

Return values:

- List of the created file names, connected with a separator each

Available since: 4.00.212

Example:

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

1.245 SetCookie function

The SetCookie function sets the value of a cookie entry. If the cookie does not yet exist, it is automatically generated. The Get/SetCookie functions are primarily intended so ELO scripting macros can store information permanently in ELO. The cookie memory is only deleted when you close ELO.

To set a cookie, you give it a name and a value. You access a cookie with its name (Ident); the value assigned to it is returned. If the cookie is unknown, an empty string is returned.

Please note that the number of cookies is limited. Each macro and any other external program should only use a few of them and each time you launch it you should use the same ones again and not keep creating new ones.

```
int SetCookie( AnsiString Ident, AnsiString Value )
```

Parameter:

- Ident Cookie name, used with GetCookie
- Value Cookie content

Return values:

- -1: Error creating cookie (e.g. memory full)
- 1: Ok

See also:

- GetCookie

1.246 SetObjAttrib function

This function sets the value of an input line in the current document form.

There are three values for each input field:

The "visible" description, shown to the user in the keywording forms as a field name (e.g. invoice number). It provides a line name for the user.

The extension, which is used in the database to identify the line in the index (e.g. RENR). It provides a line reference for the machine.

The input value; this stores the user input (e.g. 199807106)

```
int SetObjAttrib( int AttribNo, AnsiString AttribValue )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- Line 51 contains the file names of the document:
- AttribValue The new input value of this line, up to a maximum of 30 characters

Return values:

- -1: Invalid line number
- -2: No write access
- 1: Ok

See also:

- GetObjAttrib
- SetObjAttribKey
- SetObjAttribName
- SetObjAttribFlags
- SetObjAttribMin
- SetObjAttribMax
- SetObjAttribType

1.247 SetObjAttribFlags function

This function sets the flags from an input field of the current keywording form. The "Flags" value to be transmitted is an integer, the bits set there mean the following:

- Bit 0 Only accept data from keyword list
- Bit 1 Automatically add * before search term
- Bit 2 Automatically add * after search term
- Bit 3 New tab after this line
- Bit 4 Line invisible
- Bit 5 Read-only line
- Bit 6 Column with high priority, add text to the short name

```
int SetObjAttribFlags( int AttribNo, int Flags )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- Line 51 contains the file names of the document
- Flags Integer value with correspondingly set bits (see above)

Return values:

- -1: Invalid line number
- -2: No write access
- 1: Ok

Available from: 5.00.164

See also:

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

1.248 SetObjAttribKey function

This function changes the index key (group name) of a line of the current document form.

```
int SetObjAttribKey( int AttribNo, AnsiString KeyValue )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
  Line 51 contains the file names of the document
- KeyValue The new index of this line.

Return values:

- -1: Invalid line number
- -2: No write access
- 1: Ok

See also:

- SetObjAttrib
- GetObjAttribKey
- SetObjAttribName
- SetObjAttribFlags
- SetObjAttribMin
- SetObjAttribMax
- SetObjAttribType

1.249 SetObjAttribMax function

This function sets the maximum length of an input line in the current document form. This field is evaluated in the default input form within ELO. If you make your inputs using the OLE Automation program, you need to keep the maximum and minimum input lengths in mind.

```
int SetObjAttribMax( int AttribNo, int AttribMax )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- Line 51 contains the file names of the document
- AttribMax Maximum length of input, 0: No control

Return values:

- -1: Invalid input line
- -2: No write access
- 1: Ok

See also:

- SetObjAttrib
- SetObjAttribKey
- SetObjAttribName
- SetObjAttribFlags
- SetObjAttribMin
- GetObjAttribMax
- SetObjAttribType

1.250 SetObjAttribMin function

This function sets the minimum length of an input line in the current document form. This field is evaluated in the default input form within ELO. If you make your inputs using the OLE Automation program, you need to keep the maximum and minimum input lengths in mind.

```
int SetObjAttribMin( int AttribNo, int AttribMin )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- Line 51 contains the file names of the document
- AttribMin Minimum length of input, 0: No control

Return values:

- -1: Invalid input line
- -2: No write access
- 1: Ok

See also:

- SetObjAttrib
- SetObjAttribKey
- SetObjAttribName
- SetObjAttribFlags
- GetObjAttribMin
- SetObjAttribMax
- SetObjAttribType

1.251 SetObjAttribName function

This function changes the name of a line the user sees in the current document form.

There are three values for each input field:

The "visible" description, shown to the user in the keywording forms as a field name (e.g. invoice number). It provides a line name for the user.

The extension, which is used in the database to identify the line in the index (e.g. RENR). It provides a line reference for the machine.

The input value; this stores the user input (e.g. 199807106)

```
int SetObjAttribName( int AttribNo, AnsiString NameValue )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- Line 51 contains the file names of the document
- NameValue The new name of this line.

Return values:

- -1: Invalid line number
- -2: No write access
- 1: Ok

See also:

- SetObjAttrib
- SetObjAttribKey
- GetObjAttribName
- SetObjAttribFlags
- SetObjAttribMin
- SetObjAttribMax
- SetObjAttribType

1.252 SetObjAttribType function

This function sets the type of input of an input line in the current document form. This field is evaluated in the default input form within ELO. If you make your inputs using the OLE Automation program, you need to keep the type of entry in mind.

- 0: any text field
- 1: Date field
- 2: Numerical field
- 3: File number
- 4: ISO date
- 5: List entry
- 6: User field
- 7: Thesaurus
- 8: Numeric, fixed width
- 9: Numeric, fixed width, 1 decimal place
- 10: Numeric, fixed width, 2 decimal places
- 11: Numeric, fixed width, 4 decimal places
- 12: Numeric, fixed width, 6 decimal places

```
int SetObjAttribType( int AttribNo, int AttribType )
```

Parameter:

- AttribNo The lines of the input forms are numbered consecutively from 0…49.
- Line 51 contains the file names of the document
- AttribMax Permitted input type

Return values:

- -1: Invalid input line
- -2: No write access
- 1: Ok

See also:

- SetObjAttrib
- SetObjAttribKey
- SetObjAttribName
- SetObjAttribFlags
- SetObjAttribMin
- SetObjAttribMax
- GetObjAttribType

1.253 SetOption function

The SetOption function lets you change individual settings temporarily in the Options dialog box. The call gives as return value the value of the current setting.

```
AnsiString SetOption( int OptionNo, AnsiString NewValue )
```

Parameter:

- OptionNo: 0: Intray – automatically stored if index is complete
- 1: Name of fax printer (if configured)
- 2: Name of TIFF printer (if configured)
- 3: OM: name of Outlook form
- 4: OM: number of "Sender" index field
- 5: OM: number of "Recipient" index field
- 6: OM: number of "Id" index field
- 7: OM: how it is filed (0=MSG format)
- 8: Client running in an NT environment
- 9: Length of an index field
- 10: Length of short description
- 11: Length of memo text
- 12: Pre-defined Intray form
- 13: Form for new structure elements
- 14: Form for new documents
- 15: Form for new office documents
- 16: Threshold value for “large document” warning
- 17: Last ObjectId number in repository (read-only)
- 18: Duplicate checking
- 0: Normal duplicate checking
- 1: No duplicate checking, always enter
- 2: Enter first storage location as a reference
- 19: Internal default path

Return values:

- Old value of option or "ERROR"

Available since: 3.00.286; option no 3 and later: 3.00.350

Example

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

OldVal=Elo.SetOption( 0, "FALSE" )
MsgBox "The old value of the option was:" & OldVal & ", it was changed to
False."

NewVal=Elo.QueryOption(0)
MsgBox "New value of the option is set to: " &NewVal

call Elo.SetOption(0, OldVal)
```

```
MsgBox "Value was reset to original status."
```

See also:

- QueryOption

1.254 SetScriptButton function

This function lets you assign a script to a script button on the main ELO screen. The assignment is saved under the respective user.

The function is called within installation scripts, which allow scripts to be imported and buttons and menus to be assigned at the same time.

```
int SetScriptButton(int iTabSheet, int iButton, AnsiString sScriptName, int
iVisible)
```

Parameter:

- iTabSheet Select work area:
- 1: Repository work area (up to 16 buttons)
- 2: Clipboard (up to 12 buttons)
- 3: Intray (up to 12 buttons)
- 4: Search (up to 12 buttons)
- 5: Reminder (up to 12 buttons)
- iButton Number of the button (1..16 or 1..12)
- sScriptName Name of script (without extension)
- iVisible 1 = Switch button to visible
- 0 = Switch button to invisible

Starting with ELO Version 8, the ribbon is used for this.

The ribbon numbers are for the "SetScriptButton" and the names are for the "AddRibbonGroup" function.

- 1: sRibbonTabName="dxRibbon1Start";
- 2: sRibbonTabName="dxRibbon1Dokument";
- 3: sRibbonTabName="dxRibbon1Archiv";
- 4: sRibbonTabName="dxRibbon1Aufgaben";
- 5: sRibbonTabName="dxRibbon1Ansicht";
- 6: sRibbonTabName="dxRibbon1Zwischenablage";
- 7: sRibbonTabName="dxRibbon1Postbox2"; //File Intray
- 8: sRibbonTabName="dxRibbon1Suche";
- 9: sRibbonTabName="dxRibbon1Postbox2"; // Scan Intray
- 10: sRibbonTabName="dxRibbon1AufgabenTools";

The 2nd function parameter is then used for the group within the "tab".

Return values:

- -4: Script not available
- -3: Invalid value for iButton
- -2: Invalid value for Repository work area
- -1: No active work area
- 1: Ok

See also:

- SetScriptMenu
- GetScriptButton
- ImportScript

1.255 SetScriptEvent function

This function sets a script event.

```
SetScriptEvent (AnsiString Event, AnsiString Script,int Mode)
```

- Event : String with event descriptor (see Event List)
- Or position when prefixed with a # (these values may vary in future ELO versions).
-
- Script : String with script name without ending (if in default script path) or
- complete path name with filename + ending (script is then automatically
- copied to script directory)
- If an empty string is sent, the event set is deleted. In this case the mode has no significance.
-
- Mode: 0 – Change script assignment only if there is free space (if the script contains a path name,
- when it is changed an existing file is not overwritten and the script is not assigned).
-
- 1 – Always change script assignment (if script contains a path name, an existing file is overwritten when recopying)

Return values:

- -5 - Error copying script
- -4 - File name does not exist
- -3 - Script position filled
- -2 - Unknown event
- -1 - Mode not implemented
- >=0 -Event number that was set

Example:

Sets a script from the script directory in the OnTimer event if the space is not assigned

```
ELO. SetScriptEvent (“sEonTimer“, “My timer script“,0)
```

Sets a script from drive A: in the OnTimer event whether it is filled or not.

```
ELO. SetScriptEvent (“sEonTimer“, “a:\anyScript.VBS“,1)
```

Sets a script from the script directory in the event at Position 6

```
ELO. SetScriptEvent (“#6“, “Do something script“, 0)
```

Deletes a script from Event 12

```
ELO. SetScriptEvent (“#12“, ““, 0)
```

Event list:

Event	Position	Description
sEonTimer	0	On timer call
sEbeforeCollectPBox	1	Before collecting Intray entries
sEafterCollectPBox	2	After collecting Intray entries
sEbeforeCollectWv	3	Before collecting reminders
sEafterCollectWv	4	After collecting reminders
sEafterScan	5	After scanning
sEafterBarcode	6	After barcode recognition
sEafterOCR	7	After OCR
sEafterSearch	8	After search
sEafterChangeArcDepth	9	After changing repository level
sEonReworkBarcode	10	After barcode processing
sEafterMovePBox	11	After moving from the Intray
sEonClickEntry	12	When an entry is clicked
sEonWorkWv	13	When editing the keywording
sEbeforeMovePBox	14	Before moving from the Intray
sEonEnterArc	15	When entering the repository
sEonLeaveArc	16	When leaving the repository
sEonLoadSaveUser	17	When reading or saving a user
sEafterInsertNewDoc	18	After entering an item to the repository
sEafterSaveWv	19	After saving a reminder
sEbeforeDeleteWv	20	Before deleting a reminder

sEbeforeExImportEntry	21	Before exporting/importing an entry
sEbeforeShowDoc	22	Before viewing a document
sEonCheckInOutDoc	23	When checking a document in or out

Additionally available from version 4.00.190 on:

sEeditNote	24	When editing a sticky note
sEinsertRef	25	Inserting/Moving a reference
sECollectList	26	Before collecting the successors list
sEreadwriteActivity	27	Reading or writing an activity
sEviewerExport	28	Exporting in the viewer
sEbeforeSearch	29	Before the search
sEbatchStore	30	When batch filing
sEhtmlView	31	Before displaying the HTML keywording
sEdeleteEntry	33	When deleting an entry:

See also:

- GetScriptEvent

1.256 SetScriptLock function

This function locks the execution of additional script events. With this, you can suppress additional events that might be called up via their script functions. If you, for example, have script event during check in and perform own check in actions, they would in turn activate your script. The ScriptLock function was set up to avoid this type of recursion. While the script is active, no other scripts are called.

CAUTION: You should only use this function with caution and only for carefully selected functions. If you leave the lock in place, no further scripts are executed after your script.

```
int SetScriptLock( int Lock )
```

Parameter:

- Lock 1: Set lock, 0: no lock

Return values:

- Old value of the lock

Available from: 5.00.064

1.257 SetScriptMenu function

This function lets you enter an ELO script in the context menu of the main ELO screen. Each work area (Repository, Clipboard, Intray, Search and Tasks) has its own context menu. The assignment is saved under the respective user.

The function is called within installation scripts, which allow scripts to be imported and buttons and menus to be assigned at the same time.

```
int SetScriptMenu(int iTabSheet, AnsiString sScriptName)
```

Parameter:

- iTabSheet Select work area:
- 1: Repository work area
- 2: Clipboard
- 3: Intray
- 4: Search
- 5: Reminder
- sScriptName Name of script (without extension)

Return values:

- -2: Invalid value for Repository work area
- -1: No active work area
- 1: Ok

To remove a script from the context menu, simply add "1000" to the value of the "TabSheet" parameter. So, for example: ELO.SetScriptMenu 1001, "Script A"

See also:

- SetScriptButton
- GetScriptButton
- ImportScript

1.258 SetViewersReadOnly function

Use this function to set the annotation bar to 'ReadOnly" when displaying a TIFF document. With this, the user cannot affix or edit any more notes on the document.

Setting this option requires a Viewer refresh.

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

Parameter:

- ReadOnly
- 1: Turn off editing option of notes
- 2: Turn on editing option for notes
-

Return values:

- -1: No active work area
- 1: OK

1.259 ShowAutoDlg () function

Shows the dialog box created. **CreateAutoDlg must be called first.** It is not possible to call ShowAutoDlg several times in succession, because clicking OK or Cancel corrupts the information to be able to display the dialog box. So you then have to open CreateAutoDlg again.

```
int ShowAutoDlg ()
```

Return value:

- 0 – Cancel was pressed.
- 1 – OK was pressed.

Available from: Version 3.00.228

Example:

Creates an empty dialog box and displays it.

```
Elo.CreateAutoDlg ("My dialog box")
Elo.ShowAutoDlg
```

See also:

- CreateAutoDlg
- AddAutoDlgControl
- GetAutoDlgValue

1.260 ShowDocInverted function

This function shows the currently open TIFF document inverted.

```
int ShowDocInverted()
```

Parameter:

- None

Return values:

- -1: No active work area
- -2: No active viewer visible
- 1: Ok

1.261 SigId (int) property

The SigId property defines the document manager ID of the signature of a document. A wrong entry at this point leads to the destruction of the signature.

Available since: 5.00.042

See also:

- MaskKey
- DocKey
- DocKind
- DocPath

1.262 Sleep function

This function accomplishes a break with an adjustable amount of milliseconds. Thereby you can additionally execute ProcessMessages before and/or after the queue time. This request leads to the fact that you can still actively operate the Client.

```
int Sleep( int Flags, int Delay )
```

Parameter:

- Flags Bit 0: ProcessMessages before the sleep
- Bit 1: ProcessMessages after the sleep
- Rest: reserved
- Delay Queue time in milliseconds

Return values:

- 1: Ok

Available since: 4.00.054

1.263 SplitFileName function

The SplitFileName function looks at a file path (drive – path – file name or server – path – file name) and finds the path element with the drive or server, the file name or the file Id.

```
AnsiString SplitFileName( AnsiString FilePath, int iMode )
```

Parameter:

- FilePath Full file name with drive and path
- iMode 1:Returns the disk + path element
- 2: Returns the file name
- 3: Returns the file ID

Available from: 3.00.220

1.264 StartScan function

The StartScan function lets you start a scan in the Intray. With the ActionCode parameter, you can define whether a single scan or a batch scan should be performed. The second parameter determines what default scanner settings are to be used (page size etc.).

You can find the scanned image in ActivePostFile. When this entry is empty, a scanning error occurred or the user canceled the process.

```
int StartScan ( int ActionCode, int PageSize )
```

Parameter:

- ActionCode 0: No scan, just set page size
- 1: Batch scanning
- 2: Single page scan
- PageSize 0: Scan with preview
- 1..4: With scanner parameter set 1..4 (options – scanner)

Return values:

- -1: No active work area
- 1: Ok

1.265 Status function

This function sets a text in the status bar in the current main ELO window.

```
int Status( AnsiString Text )
```

Parameter:

- Text: Text to be output

Return values:

- -1: No active work area
- 1: Ok

See also:

- UpdatePostbox

1.266 StoreDirect function

This function stores the Intray entries selected in the repository, so it corresponds to the menu command 'Store directly in repository' in the Intray context menu. You need to open the desired folder in the Repository work area before running this function.

```
int StoreDirect ()
```

Parameter:

- None

Return values:

- -3: No folder selected
- -2: No entries selected in the Intray
- -1: No active work area
- 1: Ok

See also:

- StoreKeyword
- StoreMulti

1.267 StoreKeyword function

This function stores the Intray entries selected in the repository by keyword, so it corresponds to the menu command "Store keyword in repository" in the Intray context menu.

```
int StoreKeyword ()
```

Parameter:

- None

Return values:

- -2: No entries selected in the Intray
- -1: No active work area
- 1: Ok

See also:

- StoreDirect
- StoreMulti

1.268 StoreMulti function

This function files the selected entries to the repository, corresponding to the "Anonymous direct archiving" menu item in the Intray context menu.

```
int StoreMulti()
```

Parameter:

- None

Return values:

- -3: No folder selected
- -2: No entries selected in the Intray
- -1: No active work area
- 1: Ok

See also:

- StoreDirect
- StoreKeyword

1.269 TextParam (AnsiString) property

The TextParam property contains various parameters that are to be transmitted by a script event to the script or back from the script to the event. The content of this property is described by the respective script event. At all other times, this property is assigned a random value and should also not be changed.

1.270 ToClipboard function

This function sends a text to the Windows Clipboard.

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

Parameter:

- Text Text to be sent

Return values:

- Always 0

Available since: 3.00.456

Example:

```
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) ' Choose selected item
  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
```

See also:

- FromClipboard

1.271 TreeWalk function

This function runs from the start node through all child nodes and requests a script for each entry. This way you can work through whole structures with one call.

When working through child objects, the Parameters mode lets you work on the current node before the child node or after (or before and after) the child node. If you want to apply an attribute of a top-level folder to all documents, you must call it "before" (the folder attribute must be set before the child folders are edited). If you want to gather information (e.g. how many documents meet a certain criterion), then "after" is more appropriate (all the child folders must first be counted to find the total number in the folder). If the "node succession" is not important (e.g. for "Print all documents"), then you should choose the "before" method, because it puts slightly less load on the system.

The parameters mode also lets you decide whether you want to include references or stick to the main lines of succession.

With the MaxDepth parameter you can define the maximum depth you want to work on. For example, if you want to edit all folders, set the start node to 1 (repository object) and maximum depth to 2 (cabinet and folder level). Then in the callback script you only have to check whether the call refers to a cabinet object (in which case you do nothing) or a folder object. The search starts here but does not go any deeper, so this call is extremely efficient. If you want all the objects listed, you can simply set maximum depth to 15 or another sufficiently high number.

This function offers a simple means of working with whole hierarchies. It is not designed for maximum performance, because a separate script call must take place for every entry. So if you are dealing with large volumes of data or you need the fastest possible access, you must create the TreeWalk in the script itself. However, most of the time this version should be fast enough.

```
int TreeWalk( int Mode, int StartObj, int MaxDepth, AnsiString ScriptName )
```

Parameter:

- Mode Bit 0 Parent node is edited before the child nodes
- Bit 1 Parent node is edited after child node
- Bit 2 No references are included
- StartObj ELO object number of start node
- MaxDepth Maximum search depth
- ScriptName Name of callback script called for each object

Return values:

- -2: Start node not found
- -1: No active work area
- >0: Number of objects found

Starting with: 3.00.324 (Professional); 5.00.222 (Office)

Example: Script 1 called **"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 )
```

Script 2, containing the actual TreeWalk call and a list of all the short descriptions below the node currently selected.

```
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
```

1.272 UnselectAllPostBoxLines function

This function unselects all rows of the Intray.

```
int UnselectAllPostBoxLines()
```

Parameter:

- None

Return values:

- -1: No active work area
- 1: Ok

See also:

- SelectPostBoxLine
- UnselectPostBoxLine
- PostBoxLineSelected
- UnselectAllPostBoxLines

1.273 UnselectArcListLine function

This function unselects an entry in the list on the right in the Repository work area.

```
int UnselectArcListLine(int LineNo)
```

Parameter:

- LineNo Row to be unselected

Return values:

- -2: Invalid row number
- -1: No active work area
- 1: Ok

Available since: 5.00.036

Example

```
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
```

See also:

- SelectArcListLine
- ArcListLineSelected

1.274 UnselectLine function

This function unselects an entry in the current view. This function may only be used in views, where multiple selection (e.g. clipboard) is allowed. In the other work areas (such as the Repository work area), it can lead to random results.

```
int UnselectLine(int LineNo)
```

Parameter:

- LineNo Row to be unselected

Return values:

- -4: Invalid row number
- -3: No active work area
- -2: Invalid row number
- 1: Ok, was selected
- 2: Ok, was not selected

See also:

- SelectPostBoxLine
- PostBoxLineSelected
- SelectAllPostBoxLines
- UnselectAllPostBoxLines

1.275 UnselectPostBoxLine function

This function unselects an entry in the Intray.

```
int UnselectPostBoxLine(int LineNo)
```

Parameter:

- LineNo Row to be unselected

Return values:

- -2: Invalid row number
- -1: No active work area
- 1: Ok

See also:

- SelectPostBoxLine
- PostBoxLineSelected
- SelectAllPostBoxLines
- UnselectAllPostBoxLines

1.276 UnselectSearchListLine function

This function unselects one entry in the Intray.

```
int UnselectSearchListLine(int LineNo)
```

Parameter:

- LineNo Row to be unselected

Return values:

- -2: Invalid row number
- -1: No active work area
- 1: Ok

See also:

- SelectSearchListLine
- SearchListLineSelected

1.277 UpdateDocument, UpdateDocumentEx function

This function adds a new image file to an ELO document. Depending on the status of the document the old image file is overwritten (freely edited), a new version is created (version controlled) or the function is rejected (no revision allowed).

```
int UpdateDocument( int DocId, int Status, AnsiString DocFilePath )
int UpdateDocumentEx( int Id, int Status, AnsiString FilePath, AnsiString
Version, AnsiString Comment )
```

Parameter:

- DocId Internal ELO ObjectId to which new image file is to be added
- Status 1: Check access right to file path, 2: Check right to edit documents
- DocFilePath Path and name of new image file

In addition for UpdateDocumentEx:

- Version Version details, free text up to 10 characters long
- Comment Comment on new version, free text

Return values:

- -1: No active work area
- -2: Write access to document denied
- -3: DocId does not point to a document
- -4: Error reading document data from database
- -5: Error inserting image file in repository
- -6: Error entering image in database
- 1: Ok

Available from Ex-Version from 3.00.170

See also:

- InsertDocAttachment
- GetDocumentPath

1.278 UpdateNote function

The function updates a sticky note that was accessed beforehand by FindFirstNote or FindNextNote.

```
int UpdateNote()
```

Parameter:

- -

Return values:

- 1: Sticky note was updated
- -1: No active work area
- -2: No sticky note selected
- -3: Error while backing up database
- -4: Insufficient rights for writing sticky notes
- -5: The sticky note to be updated is a stamp
- -6: The sticky note to be updated is currently being edited by another user

Available since: 6.00.090

Example:

```
' Add object ID iID to the text of the first sticky
' note of a document:
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
```

See also:

- FindFirstNote
- FindNextNote
- DeleteNote

1.279 UpdateObject function

With the UpdateObject function you add an entry to the database. The object must first be created with PrepareObject or have been read from the database. If it is a new object, before saving it the system checks with IndexText whether it can identify a permitted predecessor object.

```
int UpdateObject()
```

Parameter:

- None

Return values:

- -4: Error while updating in database
- -3: Filing destination missing or unknown
- -2: Error for new entry in database
- -1: No active work area
- 1: New entry made
- 3: Update performed

See also:

- PrepareObject
- LookupIndex

1.280 UpdatePostbox function

The UpdatePostbox function triggers a repeat collection of the Intray content. In addition to the actual Intray files, TIFF printer files, network scanner files and Outlook entries are collected as well.

```
int UpdatePostbox()
```

Parameter:

- None

Return values:

- -1: No active work area
- >=0: Number of entries in the Intray

See also:

- UpdateObject
- UpdateDocument
- UpdatePostboxEx

1.281 UpdatePostboxEx function

The UpdatePostboxEx function triggers either a repeat collection of the Intray contents or you can select one specific entry from the Intray. In addition to the actual Intray files, TIFF printer files, network scanner files and Outlook entries are collected as well.

```
int UpdatePostboxEx( int iMode, int iLine )
```

Parameter:

- iMode:0: Collecting the Intray as for UpdatePostbox
- 1: Select current Intray entry by clicking icon
- 2: Select current Intray entry by clicking line of text
- 3: Select associated line number by clicking icon
- 4: Select associated line number by clicking text
- 5: Clearance for current Intray viewers

If you add a "16" to the described values, "ProcessMessages" is run internally. This leads to processing the Windows message queue

Return values:

- -1: No active work area
- >=0: Number of entries in the Intray (if collecting) or 1

See also:

- UpdateObject
- UpdateDocument

1.282 UpdateSw function

This functions changes a keyword in a keyword list. Every entry of a keyword list is given a unique indicator within its level – a 2-digit letter combination: AA, AB, AC up to ZZ. By concatenating all flags at all levels, you can specifically address every keyword in the tree. Starting with the point for the root, you can now, for example, address the first entry (AA) below the second (AB) and below that the third (AC) entry using "AAABAC".

The group designates the assignment of a list to an index field (i.e. the name of the keyword list must be the same as the group field in the keywording forms editor).

```
int UpdateSw( AnsiString Gruppe, AnsiString Parent, AnsiString Wort )
```

Parameter:

- Group Selects a keyword list
- Parent Predecessor node for the new entry
- Word New keyword

Return values:

- -1: No workspace open
- -2: Error when saving the keyword
- 1t: ok

Available from: 5.00.066

Example

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

1.283 UserFlags (int) property

The UserFlags property contains the user's rights (as a bit form). An administrator can assign any rights, while subadministrators can assign only their own rights. If subadministrators assign more rights than they actually have, there is no resulting error message; the setting is automatically adjusted.

- // Constants for rights
- #define LUR_ADMINISTRATOR 1 Main administrator
- #define LUR_EDITSYSTEM 2 Edit master data
- #define LUR_EDITARC 4 Edit repository
- #define LUR_EDITDOCS 8 Edit documents
- #define LUR_CHANGEPWD 16 Change password
- #define LUR_CHANGEREVISION 32 Change document status
- #define LUR_EDITANWENDER 64 Edit user data
- #define LUR_WFADMINISTRATE 128 Manage workflows
- #define LUR_WFSTART 0x100 Start workflows
- #define LUR_DELDOCS 0x200 Delete documents
- #define LUR_DELSORD 0x400 Delete file structure elements
- #define LUR_ARCHIVAR 0x800 !!! Is not in the rights list of ELO !!!
- #define LUR_SAPADMIN 0x1000 SAP administrator
- #define LUR_IMPORT 0x2000 Import right
- #define LUR_EXPORT 0x4000 Export right
- #define LUR_EDITMASK 0x8000 Edit keywording forms
- #define LUR_EDITSCRIPT 0x10000 Edit scripts
- #define LUR_EDITDELDATE 0x20000 Edit expiration date
- #define LUR_EDITSWL 0x40000 Edit keyword lists
- #define LUR_DELETEREADONLY 0x80000 Delete revision-controlled documents
- #define LUR_EDITREPL 0x100000 Edit replication sets
- #define LUR_CHANGEACL 0x200000 Edit access rights settings
- #define LUR_IGNOREACL 0x400000 See all entries, ignore rights settings
- #define LUR_EDITSCAN 0x800000 Edit scanner settings and profiles
- #define LUR_CHANGEMASK 0x1000000 Edit form type later on
- #define LUR_ACTPROJ 0x2000000 Create activity projects
- #define LUR_CHANGEPATH 0x4000000 Edit filing path
- #define LUR_NOLOGIN 0x8000000 Lock account
- #define LUR_DELVERSION 0x10000000 Delete versions

See also:

- ReadUser
- WriteUser
- UserParent
- UserKeys
- UserGroups
- UserFlags2

1.284 UserId (int) property

The UserId property lets you find the number of the user data record currently active. Although this property also allows write access, you should only change this number if you are certain what the result of the action will be (in general there is no reason for the script programmer to change this entry).

See also:

- UserName

1.285 UserKeys (AnsiString) property

The UserKeys property contains the user's keys (as a number sequence separated by commas, the key number paired with the access right). An administrator can assign any keys, while subadministrators can assign only their own keys. It is the responsibility of the script programmer to make sure that in this case only the available keys are assigned.

Parameter:

- Bit 1 L (Read)
- Bit 2 S (Write)
- Bit 3 D (Delete)
- Bit 4 E (Edit)

Example: 0,5,3,1

- Key no. 1 with L+D rights
- Key no. 3 with L right

See also:

- ReadUser
- WriteUser
- UserParent
- UserKeys
- UserFlags

1.286 UserName (AnsiString) property

The UserName property lets you read the user name of the user data record currently loaded. This data record may have been filled, for example, by means of an event either reading or writing it. In this case, the ActionKey property is set depending on the reason for the call (20000: immediately before restoring a user, 20001: after saving a user, 20002: after reading a user.

See also:

- UserId

1.287 UserParent (int) property

The UserParent property can assign a (sub)administrator to a user. This (sub)administrator then has the right to change the user data of the user.

See also:

- ReadUser
- WriteUser
- UserGroups
- UserKeys
- UserFlags

1.288 UserTerminate (AnsiString) property

The property contains the name of the user that has terminated this node. The name is saved in text form when the node is closed. So it is also retained if the original user is deleted or renamed in ELO Access Manager.

Available since: 3.00.508.

See also:

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

1.289 Version function

Returns the ELO version number in the form MSSBBB. (Example 123456)

- M Main version number (1 in the example)
- SS Subversion (23 in the example)
- BBB Internal build number (456 in the example)

This query lets you ascertain that your current ELO version is new enough, if you are using functions that were only added in later versions.

Warning: Starting with ELO 8, there is an 'extended' version number: Example: 8000068000

```
int Version()
```

Parameter:

- None

Return values:

- ELO version number

1.290 ViewFileName (String) property

The ViewFileName property contains the name of the document shown in the document viewer. It is usually interpreted in scripts linked to the event "Before showing document". In a script like this, you can display a different document file by changing this property.

1.291 WindowState (int)

This property lets you query the state of the ELO window.

- Values: Normal : 1
- Minimized: 2
- Maximized: 3
- Not known: -1

Available from: 3.00.218

1.292 WriteActivity (AnsiString) function

This function writes an activities entry. If the ActGuid field is empty, a new entry is created, and when a value is entered in it the data record selected is updated.

The parameter string consists of the following elements:

- ActGuid Unique ID of activity; leave emtpy for new entries!
- DocGuid GUID of ELO document (determine with GetGuidFromObj).
- Destination Recipient
- RevVers Version
- ActTStamp Timestamp, leave empty; automatically generated when saved
- Project Project name, preset if possible
- Owner Owner, ELO user number
- Creator Creator, ELO user number
- Prio 0,1 or 2
- ShortDesc Short description
- SentAt Send date in ISO format
- SentMode Send mode
- DueDate Expected return date, ISO format
- BackAt Return date, ISO format
- BackMode Return status
- Comment Comment
- FileName Name of file sent
- UD0 User-defined field 1
- UD1
- …
- UD9 User-defined field 10

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

Parameter:

- Activity: Data record to be written

Return value:

- -1 No active work area
- -2 Error writing
- 1 Ok

Available since: 3.00.360

Example:

```
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
```

See also:

- ShowActivities
- EditActivity
- ReadActivity
- WriteActivity
- NextActivity

1.293 WriteColorInfo function

This function lets you write a color definition from the current ELO color object. The color settings may be set in advance using the properties ColorInfo and ColorName.

```
int WriteColorInfo( int ColorNo )
```

Parameter:

- ColorNo: Number of color definition to be written

Return values:

- -1: No active work area
- -2: Invalid value for color number
- 1: Ok

See also:

- ReadColorNo
- ColorInfo
- ColorName

1.294 WriteKey (int KeyNo, AnsiString KeyName) function

This function creates a new key entry or renames it.

```
int WriteKey (int KeyNo, AnsiString KeyName)
```

- KeyNo: Key number starting with 1
- KeyName: Name of the key

Return values:

- -1 Invalid key number
- 0 Error
- 1 OK

Example:

Creates a new system key number 10 called accounting.

```
Elo.WriteKey (10,“Accounting“)
```

See also:

- ReadKey

1.295 WriteObjMask function

This function lets you write a form definition. The form must first have been read with ReadObjMask. If you want to create a new form, it must be initialized with ReadObjMask(9999).

```
int WriteObjMask( )
```

Parameter:

- None

Return values:

- -3: No active work area
- -2: Invalid value for form number
- -1: Error reading database
- 1: Ok

Example: (valid from Version 3.00.290)

```
' Creating a new form with an index field
NEWMASK=9999
Set Elo=CreateObject("ELO.professional")

if Elo.ReadObjMask( NEWMASK )<0 then
  MsgBox "Error preparing keywording form"
else
  Elo.ObjMName="ELO test keywording form"
  Elo.MaskFlags=25 ' Version controlled, research+file

  ' An index field, input length 5..10 characters
  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 "Error no. " & x & " while creating new keywording form."
  else
    MsgBox "New keywording form with number " & Elo.ObjMaskNo & " created."
  end if
end if
```

See also:

- ReadObjMask
- GetObjMaskNo
- MaskFlags
- SetMaskLineAcl

1.296 Function: WriteUser

The WriteUser function writes the prepared user data record back to the system file. Administrators can write any users, subadministrators can only write their own users or new users.

```
int WriteUser()
```

Parameter:

- None

Return values:

- -5: A subadministrator has attempted to write a foreign user
- -4: Error reading from AccessManager
- -3: A normal user attempted to write
- -2: Error writing to AccessManager
- -1: No user ID active

See also:

- ReadUser
- UserGroups
- UserParent
- UserKeys
- UserFlags

1.297 WriteUserProperty function

The WriteUserProperty function writes a user property in one of the eight user data fields. Remember that the first four fields are reserved by ELO (e.g. for NT name conversion). Fields 5 to 8 are freely available.

This function gives you the option of saving a property value immediately. This is particularly important if the data is to be saved by a non-administrator, in which case the WriteUser function is not available. You can achieve this by adding 1000 to the property number. If you describe the property 5, it is stored temporarily in the client and only stored permanently via WriteUser. The same property with the value 1005 saves immediately. The option is only available for properties 5…8 so a user cannot change any system settings.

```
int WriteUserProperty( int PropertyNo, AnsiString Property )
```

Parameter:

- PropertyNo Property number 1..8
- Property User data to be written

Return values:

- -1: Invalid property number
- -3: Writing directly is only permitted for properties 5..8 (1005..1008)
- -4: Error reading user data
- 1: Ok

Example:

With two simple scripts you can set ELO so that the current repository position is stored in the user data when closing. If you then access the repository again, ELO jumps to the right place.

Two scripts must be created to do this:

```
'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
```

Available from: additional option for direct saving 3.00.358

See also:

- ReadUserProperty
- UserGroups
- UserParent
- UserKeys
- UserFlags

1.298 WriteWv function

Writes a reminder date (determined by the WvIdent parameter) from the internal reminder memory to the database. This date can then be preassigned with the various property functions. WvIdent 0 causes that a new reminder date is created. Before entering a new appointment, the internal reminder memory can be deleted by calling up a ReadWv with parameter 0.

```
int WriteWv( int WvId )
```

Parameter:

- WvId: Number of appointment to be written, 0: create new date

Return values:

- -1: No active work area
- -2: Error writing
- -3: WvParent not valid
- -4: No short description entered
- 1: Ok

See also:

- ReadWv
- DeleteWv
- WvIdent
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.299 WvCreateDate (AnsiString) property

The WvCreateDate property lets you read or write the create date of a reminder date that was created. It should say the current day's date. If you leave this entry blank, it is automatically inserted when you save.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserFrom
- WvUserOwner
- WvDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.300 WvDate (AnsiString) property

The WvDate property lets you read or write the date of a reminder date.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserFrom
- WvUserOwner
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.301 WvDesc (AnsiString) property

The WvDesc property lets you read or write the memo text for a reminder date.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserFrom
- WvUserOwner
- WvCreateDate
- WvPrio
- WvParentType
- WvDate
- WvShort

1.302 WvDueDate (AnsiString) property

This property provides or sets the date of the reminder date, on which the document was seen. It must comply with the date format currently set in the system, i.e. usually in the form DD.MM.YYYY.

Maximum length: 12 characters

See also:

- WvNew
- EditWv
- WvListInvalidate
- WvActionCode

1.303 WvIdent (int) property

The WvIdent property lets you read or write the internal ELO number of the current reminder date. There is usually no reason for changing this entry, which is set with ReadWv or WriteWv.

See also:

- ReadWv
- WriteWv
- WvParent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.304 WvListInvalidate () function

This function updates the reminder list.

```
int WvListInvalidate ()
```

Return values:

- 1 Update successful
- -1 No active work area

See also:

- WvNew
- EditWv
- WvDueDate
- WvActionCode

1.305 WvNew (int) property

This property tells you if a new reminder was created.

Values:

- 0 No
- 1 Yes

See also:

- EditWV
- WvDueDate
- WvListInvalidate
- WvActionCode

1.306 WvParent (int) property

The WvParent property lets you read or write the internal ELO number of the repository entry for the current reminder. The repository entry may be a folder or a document.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.307 WvParentType (int), WvParentTypeEx (int) property

You can determine the type of ELO entry via the WvParentType or WvParentTypeEx property that is connected to the current reminder date. For all other values, this entry is determined from the database via the WvParent entry.

For repositories with a four-level hierarchy, you work with WvParentType; for those with more than four hierarchy levels you use WvParentTypeEx.

`WvParentType:`

- 1=Folder, 2=Folder, 3=Folder, 4=Document,

`WvParentTypeEx:`

- 1=Folder, 2=Folder, 3=…., …, 253=Tab, 254=Document

See also:

- ReadWv
- WriteWv
- WvIdent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvParent
- WvPrio
- WvShort
- WvDesc

1.308 WvPrio (int) property

The WvPrio property lets you read or write the priority of the current reminder date. You can choose from 3 values, 1 (important), 2 (normal) and 3 (less important). All other values are automatically set to 2 (normal priority).

See also:

- ReadWv
- WriteWv
- WvIdent
- WvUserOwner
- WvUserFrom
- WvDate
- WvCreateDate
- WvParent
- WvParentType
- WvShort
- WvDesc

1.309 WvShort (AnsiString) property

The WvShort property lets you read or write the short description of a reminder date. This input is absolutely essential for a new date. If this input is missing, the save operation is aborted with an error.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserFrom
- WvUserOwner
- WvCreateDate
- WvPrio
- WvParentType
- WvDate
- WvDesc

1.310 WvUserFrom (int) property

The WvUserFrom property lets you read or write the sender of a reminder date. This would normally be your own UserID. If you leave the entry blank, your own ID is automatically entered.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserOwner
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.311 WvUserOwner (int) property

The WvUserOwner property lets you read or write the owner of a reminder date.

See also:

- ReadWv
- WriteWv
- WvIdent
- WvParent
- WvUserFrom
- WvDate
- WvCreateDate
- WvPrio
- WvParentType
- WvShort
- WvDesc

1.312 Annex A

Description of dialog box elements in the main ELO view up to Version 7.0

If you want to work with the "ClickOn" function or superimpose scripts on standard ELO functions, you need the names of the corresponding menu commands or toolbar buttons. In the list below, we distinguish between buttons and menu commands with the codes "B:" and "M:".

The abbreviations within the identifier mean the following:

"Arc...": Repository work area, "Clip...": Clipboard, "Post...": Intray, "Search...": Search work area, "Wv...": Reminders work area, "mm...": menu item in the main menu, "...Popup...": menu item within a context menu, "Plp...": menu item within the context menu of the Intray, "...UserBtn...": script buttons

Menu text	Name of the control
B:Zoom 25%	ArcZoom25
B:Zoom 50%	ArcZoom50
B:Zoom 100%	ArcZoom100
B:Fit zoom document	ArcZoomFit
B:Back to start of repository	BtArcStart
B:Back	BtArcBack
B:Edit existing entry	BtArcEdit
B:Delete selected entry	BtArcErase
B:Suppress level change	BtArcLock
B:Fit zoom width	ArcZoomWidth
B:Print current document	BtArcPrint
B: Go to position of the last filing	BtGotoArc
B:Rotate 90° to left	ArcRot270
B:Rotate 180 degrees	ArcRot180

B:Rotate 90° to right	ArcRot90
B:Batch scan	ArcScan
B:Show information	ArcShowHint
B:User-defined zoom	ArcZoomUsr
B:Full-screen view	BtArcMaximize
B:E-mail	BtEMail
B:Fax	BtArFax
B:Zoom cursor	ArcZoom
B:OCR range	ArcZoomOcr
B:Create folder	BtNew1
B:Create folder	BtNew2
B:Add page to active repository document	ArcScanAdd
B:Check out document and edit	btCheckOut
B:Create document	BtNewDoc
B:Check in document	btCheckIn
B:Search current document	BtArcSearch
B:Fit document	ClipZoomFit
B:Zoom 100%	ClipZoom100
B:Zoom 50%	ClipZoom50
B:Zoom 25%	ClipZoom25
B:Remove from Clipboard	ClipErase
B:Fit width	ClipZoomWidth

B:Print current document	BtClipPrint
B:User-defined zoom	ClipZoomUsr
B:Rotate 90° to left	ClipRot270
B:Rotate 180 degrees	ClipRot180
B:Rotate 90° to right	ClipRot90
B:Full-screen view	BtClipMaximize
B:Fax	BtKbFax
B:Zoom cursor	ClipZoom
B:OCR range	ClipZoomOcr
B:E-mail	BtClipMail
B:Search current document	BtClipSearch
B:Zoom 25%	PostZoom25
B:Zoom 50%	PostZoom50
B:Zoom 100%	PostZoom100
B:Fit document	PostZoomAll
B:Fit width	PostZoomWidth
B:Send pages	PostSend
B:Collect Intray entries	ReloadPL
B:Delete selected entries	PostErase
B:Scan pages	PostScan
B:Join pages	PostCollapse
B:Separate pages	PostExpand

B:Select scanner	PostSelScan
B:Rotate 90° to left	PostRot270
B:Rotate 180 degrees	PostRot180
B:Rotate 90° to right	PostRot90
B:Print current document	BtPostPrint
B:Batch scan	PostDirectAppend
B:User-defined zoom	PostZoomUsr
B:Full-screen view	BtPostMaximize
B:Fax	BtPbFax
B:Zoom cursor	PostZoom
B:OCR range	PostZoomOcr
B:E-mail	BtPostMail
B:Store in repository with selection dialog box	btTreePost
B:Zoom 25%	SearchZoom25
B:Zoom 50%	SearchZoom50
B:Zoom 100%	SearchZoom100
B:Fit document	SearchZoomFit
B:Search entries	BtStartSearch
B:Fit width	SearchZoomWidth
B:Print current document	BtSrcPrint
B:Search sticky notes	BtSearchNote
B:User-defined zoom	SearchZoomUsr

B:Rotate 90° to left	SrcRot270
B:Rotate 180 degrees	SrcRot180
B:Rotate 90° to right	SrcRot90
B:Search full text database	BtSearchVt
B:Full-screen view	BtSearchMaximize
B:Fax	BtSrFax
B:Zoom cursor	SearchZoom
B:OCR range	SearchZoomOcr
B:E-mail	BtSrcMail
B:Check out document and edit	btSrCheckOut
B:Show tree view	SrcTree
B:Remove from hit list	SrcDelete
B:Check in document	btSrcCheckIn
B:Search current document	BtSearchCOLD
B:Zoom 25%	WvZoom25
B:Zoom 50%	WvZoom50
B:Zoom 100%	WvZoom100
B:Fit document	WvZoomFit
B:Show all priorities	WvPrioC
B:Show top priority only	WvPrioA
B:Show high and medium priorities	WvPrioB
B:Collect reminders	WvCollect

B:Delete date	WvErase
B:Fit width	WvZoomWidth
B:Print current document	BtWvPrint
B:User-defined zoom	WvZoomUsr
B:Rotate 90° to left	WvRot270
B:Rotate 180 degrees	WvRot180
B:Rotate 90° to right	WvRot90
B:Full-screen view	BtWvMaximize
B:Fax	BtWvFax
B:Zoom cursor	WvZoom
B:Include group dates	btWithGroup
B:Include proxy dates	btWithVert
B:Check out document	btWvCheckOut
B:Done, go on with workflow	btGoOn
B:Accept workflow	wvTakeFlow
B:Search current document	BtWvSearch
B:Options	BtOptions
B:Acknowledge messages	BtDelMsg
B:Collect messages	BtMsgReread
B:Send message	BtSendMail
B:Collect entries again	mbCollect
B:Office plans	mbNewPlan

B:Generate distribution plan	mbGeneratePlan
B:Check in document	mbCheckIn
B:Edit document	mbEdit
B:Discard document changes	mbVerw
B:Zoom cursor	CIZoomSel
B:Zoom 25%	CIZoom25
B:Zoom 50%	CIZoom50
B:Zoom 100%	CIZoom100
B:Fit width	CIZoomWidth
B:Fit document	CIZoomFit
B:Rotate 90° to left	CIRot270
B:Rotate 180 degrees	CIRot180
B:Rotate 90° to right	CIRot90
B:User-defined zoom	CIZoomUser
B:Print current document	CIPrint
B:Fax	CIFax
B:Search current document	BtCheckSearch
B:Renew document lock	mbNewCheckOut
B: Return to previous document display	mbDocBack
B: Forward to next document display	mbDocFore
M:Repository	Repository1
M:Prepare CD-ROM publication...	mmPrepCDR

M:Export...	mmExport1
M:Import...	mmImport1
M:Print repository overview...	mmPrintInfo
M:Reports	Reports1
M:View report...	mmShowReport
M:Reminder/Intray report...	Reminder report1
M:System information center...	mmInfocenter
M:System diagnosis...	mmSysDiag
M:Server status	Server status1
M:Next level	Next level1
M:Previous level	Previous level1
M:Delete/Restore entries	Deleted entries1
M:Show	mmShowDeleted
M:Restore...	mmUndelete
M:Delete permanently...	mmDropDel
M:Remove legacy documents...	mmDelDocs
M:Delete expired documents...	mmDropOld
M:Set repository keys...	mmArcKey
M:Select printer...	SelPrinter
M:Close	Close1
M:Edit	Edit1
M:Document...	mmEditDocument

M:Keywording...	mmEntryEdit
M:Select all	mmMarkAll
M:Paste to Clipboard	mmClipEntry
M:Set rights...	mmDocKey
M:Delete	mmDeleteEntry
M:Create new...	NewEntry1
M:Set font color...	mmMarkEntry
M:Reminder	mmWvEntry
M:Insert	MNInsert
M:Copy	Copy1
M:Insert link	mmAddLink
M:Collect links	mmCollectLink
M:Send documents	mmSendMap
M:Receive documents	mmReceiveMap
M:System settings	System settings1
M:Options...	mmOptions
M:Select scanner...	SelScanner
M:Select scan profile	SelPagesize
M:With preview	PgSizePrescan
M:User...	mmEditUser
M:Keywording forms...	mmEditMask
M:Document paths...	mmEditPath

M:Font color...	mmEditMarker
M:Report options...	mmEditReport
M:Key...	mmEditKey
M:File code...	mmAZDefine
M:Password...	mmEditPwd
M:Script...	Script manager1
M:Keyword lists	mmEditBuzz
M:Index fields...	mmSwlIndex
M:Version numbers...	mmSwlVer
M:Comment...	mmSwlComment
M:Templates...	mmTemplate
M:Substitute management...	mmVert
M:Encryption keys...	mmCryptParams
M:Undo postponement	WorkflowResetDelay
M:Report	mmWFRep
M:Overview	Report2
M:Document	Document1
M:Scan single pages	mmScanPage
M:Scan and join pages	mmMultiScan
M:Attach page in the repository	mmArcScanAdd
M:Insert file...	mmImportPage
M:Save file as...	GrafikExport1

M:Barcode recognition	mmBarcode
M:Insert notes	Insert notes1
M:OCR clipboard	mmOcrClip
M:Search	Eintrag1
M:Search...	mmEntrySearch
M:Full text search	mmSearchVT
M:Search for sticky note	Search for sticky note1
M:Transfer to Intray	mmToPost
M:Activate/show	mmActivateDoc
M:Check out/edit	mmEditActivateDoc
M:Print...	Print document1
M:Check in	CheckInMain
M:Send	mmSendMail
M:View	Ansicht1
M:New window	New window1
M:Close window	mmCloseView
M:Arrange freely	mmArrangeFree
M:Show side by side	mmArrangeSide
M:Tile vertically	mmArrangeTop
M:Zoom	Zoom1
M:Zoom 25%	Zoom025
M:Zoom 50%	Zoom050

M:Zoom 100%	Zoom100
M:Fit width	ZoomWidth
M:Fit height	ZoomUser
M:User-defined	ZoomUsr
M:Rotate	Rotate1
M:90°	mmRotate090
M:180°	mmRotate180
M:270°	mmRotate270
M:Full screen	Full screen1
M:View	View2
M:Repository	mmGotoArc
M:Clipboard	mmGotoClip
M:Intray	mmGotoPost
M:Search	mmGotoSearch
M:Tasks	mmGotoWv
M:Messages	mmGotoMsg
M:Paper records administration	mmGotoRecords
M:In use	mmGotoCI
M:Previous page	PagePrev
M:Next page	PageNext
M:First page	PageFirst
M:Last page	PageLast

M:Refresh	mmRefresh
M:Improve quality	mmScaleToGray
M:Sorting order	mmSortList
M:Manual	SortUser
M:Alphabetical	SortAlpha
M:Alphabetical descending	SortAlphaI
M:Filing date	SortADate
M:Filing date (inverse)	SortADateI
M:Document date	SortDDate
M:Doc. date (inverse)	SortDDateI
M:Configure toolbar...	ConfigureToolbars1
M:Help	Hilfe1
M:Use help...	HelpOnHelp
M:Contents...	HelpContent
M:About...	HelpAbout
M:Go to	SlpGoto
M:Remove from list	SlpRemove
M:Additional references	slpRefs
M:Document data	SlpDocumentData
M:Edit...	SlpEditDoc
M:Check out/edit...	mmSlpCheckOutExec
M:Activate/view...	SlpViewDoc

M:Print	SlpPrintDoc
M:Check out document	mmSlpCheckOut
M:Version history...	SlpDocHistory
M:Save file as...	SlpExport
M:Load new version from file...	SearchPopupNewVer
M:Link	SrcLink
M:Insert link	SrcMakeLink
M:Collect list	SrcCollectLink
M:Edit keywording...	SlpEdit
M:Create reminder date...	SlpMakeWv
M:Copy to Clipboard	SlpClip
M:Copy file structure	Copy file structure3
M:Insert file structure	Insert file structure3
M:Show repository entries...	Count repository entries3
M:Print document list...	SlpPrintDocList
M:Options...	SlpOptions
M:Report...	Report3
M:Multicolumn view	Multicolumn view
M:Help...	Help2
M:Delete documents	PlpEraDoc
M:Delete keywording...	PlpEraText
M:File	mmFile

M:Insert file...	PlpImportFile
M:Save file as...	PlpExportFile
M:General document template	mmGloTemplate
M:Personal document template	mmPrivTemplate
M:Edit document...	PlpEditOle
M:Create new document from template	PlpNewOle
M:Edit keywording...	PlpEditDoc
M:Rotate scanned pages	PlpRotate
M:Rotate 90°	PlpRot90
M:Rotate 180°	PlpRot180
M:Rotate 270°	PlpRot270
M:Freeze document	Freeze document1
M:Joining pages after separator pages	CollapseSeparatedDocs
M:Sort...	Sort1
M: Merge	PlpMergePages
M:Preset keywording form... :	PlpDocType
M:Store in repository with selection dialog box	PlpTreePost
M:Store automatically by filing definition...	PlpKeyArchive
M:Direct filing to repository...	PlpDirectArchive
M:Anonymous direct archiving...	PlpMultiStore
M:Attach to repository document	Attach to repository document1

M:Attach to search document	AppendSearch
M:Tie new version to repository document	plpNewVersion
M:Barcode recognition	PlpBarcode
M:Copy to other user's Intray...	PlpCopyPost
M:Send to other Intray...	PlpSendPost
M:Convert to PDF	PlpConvPDF
M:Inspect substitutes' Intrays	mmShowVertIntray
M:Help...	Help3
M:Set rights...	ArcPopupSetArcKey
M:Document data	ArcPopupDocMenu
M:Edit...	Edit2
M:Check out/edit...	mmCheckOutExec
M:Activate/view...	View1
M:Print...	Print1
M:Sort...	MNSort
M:Check out document	mmCheckOut
M:Check in document	mmCheckIn
M:Version history...	History1
M:Save file as...	ArcPopupExport
M:Load new version from file...	ArcPopupNewVer
M:Link	ArcLink
M:Insert link	ArcMakeLink

M:Collect list	ArcCollectLink
M:Set font color...	ArcPopupSetArcColor
M:Sorting	ArcPopupSetArcSort
M:Manual	AlpSortOrder
M:Alphabetical	AlpSortAlpha
M:Alphabetical descending	AlpSortInvAlpha
M:Filing date	AlpSortIDate
M:Filing date, descending	AlpSortInvIDate
M:Document date	AlpSortXDate
M:Document date, descending	AlpSortInvXDate
M:Edit keywording...	ArcPopupEditText
M:Additional references	ArcPopupRefs
M:Paste to Clipboard	ArcPopupToClip
M:Properties...	Count repository entries1
M:Copy file structure/documents	Copy file structure1
M:Insert file structure	Insert file structure1
M:General	ArcPopupGeneral
M:File current folder as a default index	NewStdReg
M:Test checksum	mmChecksum
M:Move document files	mmMoveDocs
M:Convert	Convert
M:ELO -> TIFF	Elo2Tiff

M:Outlook	mmOutlook
M:Synchronize folder contents with Outlook	DocSyncOutl
M:Transfer folder contents to Outlook	DocPopupOutl
M:Outlook folder connection...	ArcPopupOutl
M:Report...	Report1
M: Insert default index...	ArcPopupInsertStdReg
M:Full text database...	mnFulltext1
M:Retrieve full text database content (creates Intray file) ..	mnFulltextInfo
M:Reminders...	ArcPopupWv
M:File attachment	ArcPopupAttach
M:New ...	ArcPopupAddAttach
M:Activate	ArcPopupExecAttach
M:Save as...	ArcPopupSaveAttach
M:Version history...	ArcPopupAttHistory
M:Delete	ArcPopupDelAttach
M:Freeze document...	MNFreeze
M:New sticky note	ArcPopupNote
M:General sticky note...	ArcPopupStickyNote1
M:Personal sticky note...	ArcPopupPersonal
M:Stamp...	ArcPopupStamp
M:Create reminder date...	ArcPopupToWv

M:Delete...	ArcPopupRemove
M:Help...	Help4
M:Go to	WvGotoEntry
M:Edit date...	WvEditEntry
M:Delete date	WvDelEntry
M:Document data	Dokument data1
M:Edit...	WlpEditDoc
M:Check out/edit...	mmWlpCheckOutExec
M:Activate/view...	WlpViewDoc
M:Print	WlpPrintDoc
M:Check out document	mmWlpCheckOut
M:Version history...	WlpDocHistory
M:Save file as...	WvPopupSave
M:Link	WvLink
M:Insert link	WvMakeLink
M:Collect list	WvCollectLink
M:Edit keywording...	WlpEditData
M:Help...	Help5
M:Go to	ClpGoto
M:Delete from list	ClpEra
M:Copy file structure	Copy file structure2
M:Insert file structure	Insert file structure2

M:Count repository entries...	Count repository entries2
M:Document data	ClpDocData
M:Edit...	ClpEditDoc
M:Check out/edit...	mmClpCheckOutExec
M:Activate/view...	ClpViewDoc
M:Print...	ClpPrintDoc
M:Check out document	mmClpCheckOut
M:Version history...	ClpDocHistory
M:Save file as...	ClpDocExport
M:Load new version from file...	ClipPopupNewVer
M:Report...	ClpDocReport
M:Edit keywording...	ClpEdit
M:Help...	ClpHelp
M:Return to repository	KomPlpReturn
M:Delete request	KomPlpDelete
M:Edit dates	KomPlpEditDates
M:Forward to ...	KomPlpMove
M:Refuse files	KomPlpRefuse
M:Go to	cpGoto
M:Check in	cpCheckIn
M:Edit	cpEdit
M:Print	cpPrint

M:Discard	cbDel

1.313 Annex B

Description of action in the main ELO view starting with Version 8.0

Starting with version 8.0, actions are used within the program. To overlay actions with scripts or to call them up via ClickOn, you need to work with action names instead of the control names listed in appendix A starting with this version. You will find a list of action names in the later embedded PDF document, activate the document by double-clicking.

	ACT_PrintDocList	Print list
	ACT_VersionCompare	Compare versions
	ACT_BindToArchiveDoc	New version
	ACT_ForwardOneStep	Forward
	ACT_ScanAddToArchiveDoc	Attach page(s) to repository document
	ACT_SendPDF	Send as PDF
	ACT_SearchNote	Search notes
	ACT_SearchVersionComments	Search version comments
	ACT_SearchFullText	Full text search

	ACT_AMDiag	Users in system
	ACT_ServerStatus	Server status
	ACT_RepOptions	Report options
	ACT_OpenSubTree	Expand branch
	ACT_ListACLs	List permissions
	ACT_EditIndex_Popup	Keywording
	ACT_EditACL_Popup	Set permissions
	ACT_CountFolder_Popup	Count entries
	ACT_ShowDocument_Popup	Open in read-only mode
	ACT_CheckOutDoc_Popup	Check out and edit
	ACT_CheckInDoc_Popup	Check in

	ACT_ConvPDF_Popup	PDF conversion
	ACT_ConvTIFF_Popup	TIFF conversion
	ACT_NewFolder_Popup	New folder
	ACT_ClipboardCopy_Popup	Copy
	ACT_ClipboardInsert_Popup	Create reference
	ACT_ClipboardMove_Popup	Move entry
	ACT_PrintDocument_Popup	Print document
	ACT_SaveDocument_popup	Save file as
	ACT_WFStart_Popup	Start workflow
	ACT_WFAdhoc_Popup	Ad hoc workflow
	ACT_WFEntryFlowsOwnActive_Popup	My active workflows

	ACT_WFEntriesClosed_Popup	Completed workflows
	ACT_NewWv_Popup	Reminder
	ACT_ShowWvs_Popup	Reminders for entry
	ACT_Link_Popup	Link
	ACT_Delete_Popup	Delete
	ACT_BackToRoot	Home
	ACT_BackOneStep	Back
	ACT_InsertDocLeft	Document from template
	ACT_NewFolder	New folder
	ACT_EditIndex	Keywording
	ACT_CheckOutDoc	Check out and edit

	ACT_CheckinDoc	Check in
	ACT_CollectOutlook	Outlook
	ACT_SendEMail	Send document
	ACT_PrintDocument	Print document
	ACT_QuickInfo	Quickinfo
	ACT_InsertCopy	Insert
	ACT_FullScreen	Full screen
	ACT_Clipboard	Copy to Clipboard
	ACT_ShowDocument	Open in read-only mode
	ACT_DocumentVersions	Document versions
	ACT_GotoEntry	Go to

	ACT_EditDocument	Edit document
	ACT_SendLink	Send as link
	ACT_ClipboardCopy	Copy
	ACT_ClipboardInsert	Create reference
	ACT_ClipboardMove	Move entry
	ACT_Delete	Delete
	ACT_DeleteFromClipboard	Remove from Clipboard
	ACT_DeleteFromSearchList	Remove from search results
	ACT_StartSearch	Start search
	ACT_Refresh	Refresh
	ACT_ChangePassword	Change password...

	ACT_Configuration	System settings...
	ACT_Terminate	Close ELO
	ACT_Help	Help
	ACT_LoadNewDocument	Load new version
	ACT_AddPages	Add scanned pages
	ACT_CreatePreview	Create preview document
	ACT_SaveDocument	Save file as
	ACT_CreateSignature	Create signature
	ACT_CheckSignature	Check signature
	ACT_Checksum	Test checksum
	ACT_OpenAttachment	Open attachment in read-only mode

	ACT_AttachmentVersions	Attachment versions
	ACT_AddAttachment	Add attachment
	ACT_SaveAttachment	Save attachment as
	ACT_DeleteAttachment	Delete attachment
	ACT_Link	Link
	ACT_ShowReport	Repository report
	ACT_CountFolder	Count entries
	ACT_EditACL	Set permissions
	ACT_AddRegisterTemplate	Insert default index
	ACT_SaveRegisterTemplate	Save as default index
	ACT_AddDocumentFile	Insert file

	ACT_Export	Export
	ACT_Import	Import
	ACT_Unlock	Remove lock
	ACT_AddToFulltext	Add to full text database
	ACT_DeleteFromFulltext	Remove from full text database
	ACT_ShowDeleted	Show deleted entries
	ACT_Undelete	Restore
	ACT_DieHard	Permanently remove deleted entries
	ACT_SearchOptions	Search options
	ACT_ExtendedSearch	Search keywording
	ACT_WvPrioA	Priority A

	ACT_WvPrioB	Priority B
	ACT_WvPrioC	Priority C
	ACT_NewWv	Reminder
	ACT_EditWv	Edit task
	ACT_DeleteWv	Delete reminder
	ACT_ScanSingle	Scan pages
	ACT_ScanMultiple	Scan document
	ACT_CollapsePages	Join pages
	ACT_CollapseSeparationSheet	Join (separator pages)
	ACT_SeparatePages	Split pages
	ACT_PostboxInsertPage	Insert file

	ACT_PostboxEditIndex	Keywording
	ACT_PostboxDeleteIndex	Delete keywording
	ACT_Barcode	Barcode recognition
	ACT_PostboxDelete	Delete
	ACT_MoveToArchive	File to repository
	ACT_MoveToArchiveAuto	Automatic filing
	ACT_AddPagesPostbox	Add pages
	ACT_EditScanProfiles	Scan profiles
	ACT_SelectScannerPostbox	Select scanner
	ACT_OptUser	User
	ACT_OptMasks	Keywording forms

	ACT_OptDocumentPaths	Document paths
	ACT_OptColorConfig	Font color
	ACT_OptReport	Activate report
	ACT_OptKeys	Keys
	ACT_OptProjects	Activity projects
	ACT_OptScripts	Scripts
	ACT_OptOptions	Configuration...
	ACT_AboutELO	About...
	ACT_FaxDocument	Fax document
	ACT_Thumbnails	Thumbnail view
	ACT_DoubleView	Visual comparison

	ACT_ReScan	Rescan page
	ACT_SendFromPostbox	Move to other user's Intray
	ACT_ShowSearchTree	Show tree view
	ACT_PrepareDVDOutput	Read-only copy of the repository
	ACT_WVReport	Reminders overview
	ACT_ConvPDF	PDF conversion
	ACT_ConvTIFF	TIFF conversion
	ACT_TwinFree	Arrange freely
	ACT_TwinSideBySide	Show side by side
	ACT_TwinOverUnder	Show stacked
	ACT_SlideShow	Start slide show

	ACT_MarkAll	Select all
	ACT_ListToSearch	Move list to search area
	ACT_DocToPostbox	Copy to Intray
	ACT_NavigateToTemplateFolder	Templates
	ACT_ShowTIFFPreview	Show preview document
	ACT_BuzzwordsGlobal	Global keyword list
	ACT_BuzzwordsVersionComment	Version comment keyword list
	ACT_BuzzwordsVersionNumbers	Version number keyword list
	ACT_Cancel	Cancel
	ACT_SysInfoCenter	System information center
	ACT_SystemDiag	System diagnostics

	ACT_PrintArchive	Print repository summary
	ALT_CheckDocManager	Check repository contents
	ACT_DelVersions	Remove deleted versions
	ACT_DelOldDocs	Remove old documents
	ACT_DelVerfallsDocs	Delete expired documents
	ACT_OnlineFAQ	Online FAQ
	ACT_ScanToArchive	Scan to repository
	ACT_MultiColumnSearchList	Multicolumn view
	ACT_VoiceInfo	Voice note
	ACT_FullTextContent	Show full text database content
	ACT_RotateSort	Rotate/Sort

	ACT_CopyToOtherPostbox	Copy to other user's Intray
	Act_SaveAnonym	Batch filing
	ACT_CrankPages	Merge pages
	ACT_ListExcel	Output list to Excel
	ACT_ELOConnector	Connector
	ACT_NewActivity	Activity
	ACT_DailyRet	Daily returns
	ACT_DeadLines	Overdue tasks
	ACT_ShowWvs	Reminders for entry
	ACT_RegisterActivity	Register for monitoring
	ACT_ReturnDocument	Return

	ACT_DocumentActivities	Activities for entry
	ACT_BindOutlook	Add to Outlook
	ACT_SyncOutlook	Synchronize
	ACT_ToOutlook	Move documents
	ACT_EditCryptParams	Encryption keys
	ACT_MovoToSelectedArchiveReg	Direct filing
	ACT_CollectOutlook2	File Outlook folder
	ACT_CheckUpdate	Check for updates
	ACT_ConfigConnector	Configure Scan&Archive|15
	ACT_MarginNote1	New margin note
	ACT_MarginNote2	Personal margin note

	ACT_MarginNote3	Permanent margin note
	ACT_MarginNote1a	General margin note
	ACT_ReportOptions	Report options
	ACT_DiscardChanges	Discard document changes
	ACT_PreprocessPostbox	OCR preprocessing
	ACT_MoveDocuments	Move document files
	ACT_ShowEntryReport	Report for entry
	ACT_UserFeedback	User feedback
	ACT_ConfSubstitutes	Substitution rules
	ACT_BuzzwordsWorkflow	Workflow keyword list
	ACT_TranslationMap	Translation table

	ACT_WorkflowTemplates	Workflow templates
	ACT_ConfAZ	File code
	ACT_EditReplZones	Replication sets
	ACT_AMKey	Repository key
	ACT_CheckACLs	Check permissions
	ACT_CheckCharSet	Character set check
	ACT_CycleCheck	Cycle check
	ACT_WFCancel	Cancel postponement
	ACT_WFDefer	Postpone workflow
	ACT_WFDelegate	Delegate workflow
	ACT_WFStart	Start workflow

	ACT_WFActiveShow	Workflow overview
	ACT_WFAccept	Accept workflow
	ACT_WFConfirm	Pass workflow forward
	ACT_WFRelease	Hand off workflow
	ACT_WFAdhoc	Ad hoc workflow
	ACT_WFEdit	Show workflow
	ACT_WFEntryFlowsActive	Active workflows
	ACT_WFReject	Return workflow
	ACT_WFSetSubstitute	Assign substitute
	ACT_NewsMsg	New message
	ACT_FilterReplSets	Replication set filter

	ACT_WFEntryFlowsOwnActive	My active workflows
	ACT_WFEntriesClosed	Completed workflows
	ACT_WorkflowsActiveDialog	Workflows for processing
	ACT_WFEscalation	Passed workflow deadlines
	ACT_ShowVertPostbox	Inspect substitutes' Intrays
	ACT_AssignReplSets	Assign replication set
	ACT_TasksToExcel	Output list to Excel
	ACT_TasksToHTML	List output HTML
	ACT_WFEntriesAllActive	Show active workflows
	ACT_Backup	Backup
	ACT_FTService	Full text service

	ACT_SrcToExcel	Output list to Excel
	ACT_SrcToHTML	List output HTML
	ACT_SearchBin	Search deleted entries
	ACT_BatchEdit	Batch keywording
	ACT_HorizontalSplit	ACT_HorizontalSplit
	ACT_VerticalSplit	ACT_VerticalSplit
	ACT_WvSortorder	Sorting order
	ACT_WvCollapseExpand	Expand/collapse
	ACT_iSDate	Date
	ACT_iSForm	Form
	ACT_iSType	Object type

	ACT_iSOwner	Filed by
	ACT_iSAddField	Index field
	ACT_Undo	Undo