OLE_Server

COLLABORATORS			
	TITLE : OLE_Server		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 24, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

OLE_Server

1.1 OLE Server - Technical Informations

```
The OLE server - v1.13 - Internal structure and implementation.
 First of all
 Server lists
 Localization, configuration, message port
 Global variables, GUI lists
 Server control window
 Main loop and commands execution
 Free all, clean all and exit

Server ARexx commands
 ABOUT
 COMPLETE
 CONFIG
 ERROR
 ICONIFY
 INFO
 NEWJOB
 NEWPREFS
 QUIT
 SETJOB
 UNICONIFY
 WINDOW
```

1.2 first of all

```
/*
 *  These strings will be used in most requesters.
 */
ver_TAG = "$VER: OLE_Server 1.13 (10.Feb.1995)"
tit_TAG = ' OLE Server'

CALL  IniLibs ()

/*
 *  Bring up a message to inform the user that the server
```

```
 *  is starting correctly.
 */
CALL PostMsg(100,100,'\' || LEFT(tit_TAG || ':',40) || '\')
```

1.3 server lists

```
/*
 *  These lists are used by the server in order to take track of
 *  the execution of each  job .
 *
 *  module.0      = max number of jobs currently running
 *  module.jobID.0    =  modules  in the job number jobID
 *  module.jobID.modID  = file name for the module modID in the job jobID
 *  module.path     = path where all OLE modules are stored
 */
module.  = ''; module.path = 'OLE:'

/*
 *  Each module may have more than one startup mode.
 *
 *  status.jobID.modID  = startup mode (default = '')
 */
status.  = ''

/*
 *  Each module may have a localization file and/or a particular
 *  user configuration. These lists are initialized by the appropriate
 *  procedures:  ReadLocale () and  ReadConfig ().
 *
 *  locale.path, config.path = path for localization and configuration files
 */
locale.  = ''; locale.path = 'OLE:Catalogs/'
config.  = ''; config.path = 'OLE:Prefs/'

/*
 *  Each job gets a set of objects:
 *  An IDCMP used to put out a GUI, a clip used to pass arguments from
 *  the server and a module, a pipe channel used for datas interchange
 *  between two modules.
 *  In any case the olepipe may be a clip with the same function.
 *  In addition each job has own userport and userscreen information.
 *
 *  olewin.jobID    = IDCMP for the job number jobID
 *  olewin.jobID.0    = modID of the window owner for job jobID
 */
olewin.  = 'OLE_WIN'    /*
oleclip. = 'OLE_CLIP'    *  default values without indexes
olepipe. = 'OLE_PIPE'    */

/*
 *  Each module may open a message port where will arrive messages
 *  either from the GUI or from another module.
 *
 *  oleport.jobID.modID = message port for module.jobID.modID
 */
```

```
oleport. = 'OLE_HOST'

/*
 *  This list record the progress indicator status for each job.
 */
oleind.  = 0

/*
 *  These lists store the caller environment, for each job.
 */
userport.   = 'REXX'
userscreen. = 'Workbench'
```

1.4 localization, configuration, message port

```
/*
 *  In order to have the same procedures that manage all lists,
 *  the OLE server itself is treted as a module.
 *
 *  The server process has jobID = 0 and modID = 1.
 *  The following lines take care of the server correct start up.
 */
module.0  = 0
module.0.0  = 1
module.0.1  = 'OLE.rexx'
olewin.0  = 'OLE_IDCMP'
oleport.0.1 = 'OLE_SERVER'

/*
 *  Load the server localization and configuration files
 */
CALL  ReadLocale (0,1)
CALL  ReadConfig (0,1)
CALL PostMsg(,,'\' || tit_TAG || GetLocale(0,1,8))

/*
 *  Now we can try to open the server message port
 */
IF ~OPENPORT(oleport.0.1) THEN DO
  CALL PostMsg()
  /*
   *  In case of OPENPORT() fails, a requester will inform the user
   *  before exiting.
   */
  CALL RTezRequest( GetLocale (0,1,'ERR_1',oleport.0.1),GetLocale(0,1,'OK3'), ↩
     tit_TAG)
  EXIT 20
  END
```

1.5 global variables, gui lists

```
/*
 *  Read the variables stored in the configuration file
 */
CALL  GetServerConfig ()

/*
 *  When a module ask the server to open a new window,
 *  these lists store the given parameters.
 *  These lists are usefull for commands like UNICONIFY
 *  that produce temporary modifications in the modules windows.
 *
 *  winw.jobID  = window width for the job jobID
 *  winh.jobID, winl.jobID, wint.jobID, boxw.jobID, boxh.jobID
 *  are, in order, window height, left corner, top corner,
 *  width and height for the gadgets area.
 */
winw. = 120; winh. = win.bt; winl. = 130; wint. = 0
boxw. = 0; boxh. = 0

/*
 *  We could decide that our progress indicator is too small.
 *  Changing this value in the right range, we will be able to make
 *  it heigher.
 */
oleind.hei = 20     /* min = 18  max = 100 */

/*
 *  Each job may open windows with custom flags sets
 *
 *  idcmp.0 and flags.0 are reserved for the server
 */
idcmp. = "MENUPICK CLOSEWINDOW GADGETUP"
flags. = "WINDOWDRAG WINDOWDEPTH WINDOWCLOSE ACTIVATE"
```

1.6 server control window

```
/*
 *  This are the server window parameters
 */
idcmp.0.1 = "MENUPICK"
flags.0.1 = "WINDOWDRAG WINDOWDEPTH"

/*
 *  Start the intuition host for the OLE Server,
 *  in case of failure exit.
 */
CALL PostMsg(,,GetLocale(0,1,9))
IF ~ NewWindow (0,1,tit_TAG) THEN DO
  CALL PostMsg()
  EXIT 10
  END
```

1.7 main loop and commands execution

```
CALL PostMsg()
rt_TAG  = 'rtez_flags=ezreqf_centertext'
/*
 *  Here is the main loop. Here the OLE server wait for commands either from
 *  the OLE startup scripts and from modules. Every message just arrived at
 *  the server port is parsed to extract commands and their parameters.
 *  Each commands is executed before returning control to the caller.
 *
 *  The end-of-loop condition is the command 'QUIT'
 */
DO UNTIL cmd = 'QUIT'

  CALL WAITPKT(oleport.0.1)
  pkt = GETPKT(oleport.0.1)

  IF pkt == NULL() THEN ITERATE

  /*
   *  Extract command string, jobID, modID and arguments.
   *  Execute the command.
   */
  PARSE VALUE GETARG(pkt) WITH cmd jobID modID argv
  SELECT

     COMPLETE
     SETJOB
     ICONIFY
     UNICONIFY
     WINDOW
     INFO
     CONFIG
     NEWJOB
     NEWPREFS
     ABOUT
     ERROR
     QUIT

    OTHERWISE DO

      cmd = GETARG(pkt,0)
      DO i = 1 TO 15
        cmd = cmd || '0A'x || GETARG(pkt,i)
      END
      CALL RTezRequest( GetLocale (0,1,'ERR_3',cmd),GetLocale(0,1,'OK3'),tit_TAG)
    END
  /*
   *  end of SELECT
   */
  END
  CALL REPLY(pkt,0)
/*
 *  end of DO
 */
END
```

1.8 free all, clean all and exit

```
/*
 *  free all user jobs, closing all ports, deleting all...
 */
DO i = 1 TO module.0
  IF module.i.0 ~= '' THEN CALL  FreeJob (i)
END

/*
 *  Close the server message port
 */
CALL CLOSEPORT(oleport.0.1)

CALL FreeJob(0)

EXIT 0
```

1.9 getlocale()

```
/*
 *  string = GetLocale(jobID,modID,stringID,...)
 *
 *  This procedure differ from that used for all others modules because
 *  of its ability to read from every localization clip.
 *
 *  In addition here I can use preformatted strings:
 *
 *    text = GetLocale(0,1,'ERR_1',oleport.3.2)
 *
 *    the procedure search for a message associated with strID=ERR_1,
 *    if there is a "%s" in it then the procedure will replace that
 *    symbol with the string contained in the variable oleport.3.2;
 *    each occurrance of the symbol "%s" will be replaced with a given
 *    string in the argument list.
 *
 *  "Hello, %s!!!" + "World" ==> "Hello, World!!!"
 */
GetLocale: PROCEDURE EXPOSE locale.
ARG jobID,modID,strID

  IF strID = '' THEN RETURN ''
  strID = 'þ'strID'þ'; PARSE VALUE GETCLIP(locale.jobID.modID) WITH (strID)text'Þ'

  DO i = 4
    PARSE VAR text text '%s' clip
    IF clip = '' THEN BREAK
    text = text || ARG(i) || clip
  END

RETURN text
```

1.10 drawborder()

```
/*
 *  DrawBorder(x1,y1,x2,y2,type)
 *
 *  This procedure is used to build the progress indicator, drawing a 3D
 *  border in a module window.
 *
 *  x1, y1, x2, y2 are all referred to the gadgets area of the window
 *  type = 0,1,2 for NONE, OUT, IN
 */
DrawBorder:

  IF ARG(5) = 0 THEN RETURN

  x1 = win.bl + ARG(1); y1 = win.bt + ARG(2)
  x2 = win.bl + ARG(3); y2 = win.bt + ARG(4)
  CALL Move(olewin.jobID,x1,y2)
  CALL SetAPen(olewin.jobID,3 - ARG(5))
  CALL Draw(olewin.jobID,x1,y1); CALL Draw(olewin.jobID,x2,y1)
  CALL SetAPen(olewin.jobID,ARG(5))
  CALL Draw(olewin.jobID,x2,y2); CALL Draw(olewin.jobID,x1,y2)

RETURN
```

1.11 setoleind()

```
/*
 *  percent = SetOleInd(percent)
 *
 *  procedure to draw and update a progress indicator
 */
SetOleInd:

  x1 = win.bl + 12; y1 = win.bt + 8
  x2 = boxw.jobID - 24; y2 = win.bt + oleind.hei - 8
  CALL SetAPen(olewin.jobID,0)
  CALL RectFill(olewin.jobID,x1,y1,x1 + x2,y2)
  CALL SetAPen(olewin.jobID,oleind.color)
  CALL RectFill(olewin.jobID,x1,y1,x1 + ARG(1) * x2 % 100,y2)

RETURN ARG(1)
```

1.12 readlocale()

```
/*
 *  ReadLocale(jobID,modID)
 *
 *  This procedure load the localization file of a module and
 *  store it in the clipboard area.
 *
 *  The english localization is always loaded to provide those string
```

```
 *  not translated yet from other languages.
 */
ReadLocale: PROCEDURE EXPOSE module. locale.
ARG jobID,modID

  /*
   *  localization file name
   */
  locale = module.jobID.modID || '.catalog'
  /*
   *  localization clip name, this will contain all strings
   */
  locale.jobID.modID = locale || '_' || jobID

  clip = ''
  IF OPEN(loc,locale.path || 'english/' || locale,'R') THEN DO
    /*
     *  The value of 20000 is used to read the entire file.
     *  Don't forget that the largest clip must be of 64k.
     */
    clip = READCH(loc,20000)
    CALL CLOSE(loc)
    END

  IF GETENV('language') ~= 'english' THEN
    IF OPEN(loc,locale.path || GETENV('language') || '/' || locale,'R') THEN DO
      clip = READCH(loc,20000) || clip
      CALL CLOSE(loc)
      END

  CALL SETCLIP(locale.jobID.modID,clip)

RETURN
```

1.13 readconfig()

```
/*
 *  ReadConfig(jobID,moduleID)
 *
 *  This procedure load the configuration file of a module and store
 *  it in the clipboard area.
 */
ReadConfig: PROCEDURE EXPOSE module. config.
ARG jobID,modID

  /*
   *  configuration file name
   */
  config = module.jobID.modID || '.prefs'
  /*
   *  configuration clip name
   */
  config.jobID.modID = config || '_' || jobID

  IF OPEN(cfg,config.path || config,'R') THEN DO
```

```
    CALL SETCLIP(config.jobID.modID,READLN(cfg))
    CALL CLOSE(cfg)
    END

RETURN
```

1.14 writeconfig()

```
/*
 *  WriteConfig(jobID,modID)
 *
 *  Every module update its configuration clip when the user change
 *  anything. At any time user can save the new configuration.
 *  This procedure take the configuration clip of the given module
 *  and save it in the appropriate file.
 */
WriteConfig: PROCEDURE EXPOSE module. config. locale.
ARG jobID,modID

  IF ~SHOW('C',config.jobID.modID) THEN RETURN

  /*
   *  configuration file name
   */
  config = module.jobID.modID || '.prefs'

  IF OPEN(cfg,config.path || config,'W') THEN DO
    CALL WRITELN(cfg,GETCLIP(config.jobID.modID))
    CALL CLOSE(cfg)
    END

  /*
   *  inform the user in case of OPEN() fail
   */
  ELSE CALL RTezRequest( GetLocale (0,1,'ERR_4',config),GetLocale(0,1,'OK4'), ↩
     tit_TAG)

RETURN
```

1.15 freejob()

```
/*
 *  FreeJob(jobID)
 *
 *  This procedure remove a job from the server lists
 */
FreeJob:
ARG jobID

  /*
   *  Quit the IDCMP of the given job
   */
```

```
  IF SHOW('P',olewin.jobID) THEN CALL Quit(olewin.jobID)

  /*
   *  Free all objects allocated for each module and send the 'QUIT'
   *  command to all still running.
   */
  DO i = 1 TO module.jobID.0

    IF SHOW('P',oleport.jobID.i) THEN INTERPRET 'ADDRESS' oleport.jobID.i 'QUIT'

    CALL SETCLIP(locale.jobID.i,'')
    CALL SETCLIP(config.jobID.i,'')
  END

  /*
   *  Is not necessary to reset all variables,
   *  we don't care about winw.jobID ...
   */
  module.jobID.0 = ''
  CALL SETCLIP(oleclip.jobID,'')

RETURN
```

1.16 inimodule()

```
/*
 *  IniModule(jobID,modID)
 *
 *  prepare the execution of a new module
 *  free an entire job if it is terminated
 */
IniModule:
ARG jobID,modID

  IF module.jobID.modID = '' THEN DO
    CALL  FreeJob (jobID)
    RETURN
    END

  IF SHOW('P',oleport.jobID.modID) THEN RETURN

  CALL  ReadLocale (jobID,modID)
  CALL  ReadConfig (jobID,modID)

  clip = jobID modID ,
         win.bl win.bt win.fontw win.fonth ,
         olewin.jobID oleport.0.1 oleport.jobID.modID ,
         userscreen.jobID userport.jobID olepipe.jobID ,
         locale.jobID.modID config.jobID.modID status.jobID.modID

  CALL  RunModule (module.path || module.jobID.modID,oleclip.jobID,clip)

RETURN
```

1.17 runmodule()

```
/*
 *  launch the execution for the given external module
 *
 *  RunModule(modulename,startupclipname,clipcontent)
 */
RunModule:

  CALL SETCLIP(ARG(2),ARG(3))
  ADDRESS COMMAND 'Run >NIL: Rx "CALL' "'" || ARG(1) || "'" || '(' || ARG(2) || ') ↩
     "'

RETURN
```

1.18 newwindow()

```
/*
 *  procedure to open a new window
 *
 *  BOOLEAN = NewWindow(jobID,modID,title)
 */
NewWindow:
ARG jobID,modID

  /*
   *  create a new intuition host if it doesn't exists
   */
  IF ~SHOW('P',olewin.jobID) THEN DO
    CALL  RunModule ('New_Host.rexx',oleclip.jobID,olewin.jobID','oleport.0.1',' ↩
       userscreen.jobID',')
    IF ~ Wait _For_Port(olewin.jobID) THEN DO
      CALL RTezRequest( GetLocale (0,1,'ERR_2',olewin.jobID),GetLocale(0,1,'OK2'), ↩
         tit_TAG)
      RETURN 0
      END

      /*
       *  adjust menu background and borders colors
       */
      CALL SetReqColor(olewin.jobID,"BLOCKPEN",2)
      CALL SetReqColor(olewin.jobID,"DETAILPEN",1)
    END

  ELSE CALL CloseWindow(olewin.jobID,"CONTINUE")

  CALL OpenWindow(olewin.jobID,winl.jobID,wint.jobID,winw.jobID,winh.jobID,idcmp. ↩
     jobID.modID,flags.jobID.modID,ARG(3))
  CALL SetFont(olewin.jobID,win.font,win.fonth)
  CALL SetDrMd(olewin.jobID,'JAM1')
  CALL AddMenu(olewin.jobID,ARG(3))

  olewin.jobID.0 = modID
  IF jobID = 0 THEN DO
```

```
    CALL AddItem(olewin.jobID,GetLocale(0,1,1),'')
    CALL AddSubItem(olewin.jobID,GetLocale(0,1,2),'ABOUT' 0 1 'OLE','o')
    CALL AddSubItem(olewin.jobID,GetLocale(0,1,4),'ABOUT' 0 1 'ABOUT','a')
    CALL AddItem(olewin.jobID,'New Prefs','NEWPREFS','n')
    CALL AddItem(olewin.jobID,GetLocale(0,1,12),'NEWJOB %1' || userport.0 || '%2' ↩
       || userscreen.0 || '%3Config.ole%4CatCompiler.ole','c')
    CALL AddItem(olewin.jobID,GetLocale(0,1,7),'QUIT','q')
    END

  ELSE DO
    CALL SetNotify(olewin.jobID,"CLOSEWINDOW",oleport.0.1)
    CALL ModifyHost(olewin.jobID,"CLOSEWINDOW",'SETJOB' jobID 'end')
    CALL SetNotify(olewin.jobID,"GADGETUP",oleport.jobID.modID)
    CALL SetNotify(olewin.jobID,"MENUPICK",'REXX')
    cmd = "CALL AddItem(" || olewin.jobID || ",'" || GetLocale(0,1,3) || "'," ||  ↩
       '2227'x || ADDRESS oleport.0.1 'ABOUT' jobID modID 'ABOUT' || '2722'x || ") ↩
       "
    cmd = cmd "; CALL AddItem(" || olewin.jobID || ",'" || GetLocale(0,1,5) ||  ↩
       "'," || '2227'x || ADDRESS oleport.0.1 'CONFIG' jobID modID || '2722'x ||  ↩
       ",'s')"
    cmd = cmd "; CALL AddItem(" || olewin.jobID || ",'" || GetLocale(0,1,6) ||  ↩
       "'," || '2227'x || ADDRESS oleport.0.1 'ICONIFY' jobID modID '%f %e' ||  ↩
       '2722'x || ",'i')"
    cmd = cmd "; CALL AddItem(" || olewin.jobID || ",'" || GetLocale(0,1,7) ||  ↩
       "'," || '2227'x || ADDRESS oleport.jobID.modID 'QUIT' || '2722'x || ",'q')"
    INTERPRET cmd
    CALL ScreenToFront(userscreen.jobID)
    END

RETURN 1
```

1.19 inilibs()

```
/*
 *  These procedure load external libraries used by all modules.
 *
 *  The "rexxarplib.library" is the GUI point of support,
 *  all requester use, instead, the "rexxreqtools.library".
 *  All messages ports are created and managed by the "rexxsupport.library"
 */
IniLibs: PROCEDURE

  /*
   *  an higher priority for the 'rexxarplib.library'
   *  to speed up all operations involving gadgets
   */
  pri.1 = 5; lib.1 = 'rexxarplib.library'
  pri.2 = 0; lib.2 = 'rexxreqtools.library'
  pri.3 = 0; lib.3 = 'rexxsupport.library'

  /*
   *  in case ADDLIB() fails, simply exit
   */
  DO i = 1 TO 3
    IF (ADDLIB(lib.i,pri.i,-30,0) | SHOW('L',lib.i)) = 0 THEN EXIT 20
```

```
  END

RETURN
```

1.20 getserverconfig()

```
/*
 *  To build correctly a GUI, we need some parameters to know
 *  dimensions of window borders, the font we will use and its size,
 *  something about the progress indicator.
 */
GetServerConfig:

  /*
   *  The server, like each module, has a default configuration
   *  stored internally.
   */
  IF ~SHOW('C',config.0.1) THEN DO
    win.bl = 3; win.bt = 15; win.br = 4; win.bb = 2
    win.font = 'topaz.font'; win.fonth = 8; win.fontw = 8
    oleind.outer = 0; oleind.inner = 0; oleind.color = 4
    END

  /*
   *  If the user has saved his own configuration this will be used
   *  instead.
   */
  ELSE PARSE VALUE GETCLIP(config.0.1) WITH win.bl','win.bt','win.br','win.bb',' ↩
     win.font','win.fonth','win.fontw','oleind.outer','oleind.inner','oleind.color ↩
     ','

  delaytime = 2 /* secs */

RETURN
```

1.21 complete

```
/*
 *  COMPLETE jobID modID percent
 *
 *  This command start or update the progress indicator
 *  for the job jobID.
 */
WHEN cmd = 'COMPLETE' THEN DO

  IF oleind.jobID = 0 THEN DO
    CALL CloseWindow(olewin.jobID,"CONTINUE")
    CALL OpenWindow(olewin.jobID,winl.jobID,wint.jobID,winw.jobID,oleind.hei + win ↩
       .bt + win.bb,idcmp.0,flags.0,GetLocale(jobID,modID,'TITLE'))
    CALL  DrawBorder (2,2,boxw.jobID - 2,oleind.hei - 2,oleind.outer)
    CALL DrawBorder(10,6,boxw.jobID - 10,oleind.hei - 6,oleind.inner)
    END
```

```
  oleind.jobID =  SetOleInd (argv)
END
```

1.22 setjob

```
/*
 *  SETJOB jobID modID
 *  SETJOB jobID 'end'  terminate the job execution
 */
WHEN cmd = 'SETJOB' THEN
  CALL  IniModule (jobID,modID)
```

1.23 iconify

```
/*
 *  ICONIFY jobID modID
 */
WHEN cmd = 'ICONIFY' THEN DO
  PARSE VAR argv winl.jobID wint.jobID .
  CALL CloseWindow(olewin.jobID,"CONTINUE")
  CALL OpenWindow(olewin.jobID,winl.jobID,0,winw.jobID,win.bt,idcmp.0,flags.1,  ↩
     GetLocale (jobID,modID,'TITLE'))
  CALL SetNotify(olewin.jobID,"CLOSEWINDOW","REXX")
  INTERPRET "CALL ModifyHost(" || olewin.jobID || ",'CLOSEWINDOW'," || '2227'x ||  ↩
     ADDRESS oleport.0.1 'UNICONIFY' jobID modID ';' ADDRESS oleport.jobID.modID ' ↩
     UNICONIFY' || '2722'x || ")"
END
```

1.24 uniconify

```
/*
 *  UNICONIFY jobID modID
 */
WHEN cmd = 'UNICONIFY' THEN
  IF ~ NewWindow (jobID,modID, GetLocale (jobID,modID,'TITLE')) THEN
    CALL  FreeJob (jobID)
```

1.25 window

```
/*
 *  WINDOW jobID modID width height nidcmp nflags
 */
WHEN cmd = 'WINDOW' THEN DO
  PARSE VAR argv boxw.jobID boxh.jobID a1 a2 .

  winw.jobID = boxw.jobID + win.bl + win.br
```

```
  winh.jobID = boxh.jobID + win.bt + win.bb
  winl.jobID = (ScreenCols(userscreen.jobID) - winw.jobID) % 2
  wint.jobID = (ScreenRows(userscreen.jobID) - winh.jobID) % 2
  oleind.jobID = 0

  idcmp.jobID.modID = idcmp.a1; flags.jobID.modID = flags.a2

  IF ~ NewWindow (jobID,modID, GetLocale (jobID,modID,'TITLE')) THEN CALL  FreeJob ↩
      (jobID)
END
```

1.26 info

```
/*
 *  INFO jobID modID request
 */
WHEN cmd = 'INFO' THEN
  INTERPRET 'CALL SETCLIP(' || oleclip.jobID || ',' || argv || ')'
```

1.27 config

```
/*
 *  CONFIG jobID modID
 */
WHEN cmd = 'CONFIG' THEN
  CALL  WriteConfig (jobID,modID)
```

1.28 newjob

```
/*
 *  CALL SendParsed('OLE_SERVER','NEWJOB',userport,userscreen,mod1,mod2,...)
 */
WHEN cmd = 'NEWJOB' THEN DO
  /*
   *  set the jobID for the new job
   *  update max number of jobs in the server lists
   */
  DO jobID = 1 UNTIL module.jobID.0 = ''; END
  module.0 = MAX(module.0,jobID)
  /*
   *  read all modules for this job
   *  separate the module name from its status, if present
   */
  DO i = 1 WHILE GETARG(pkt,i + 2) ~= ''
    oleport.jobID.i = oleport. || '.' || jobID || '.' || i
    PARSE VALUE GETARG(pkt,i + 2) WITH module.jobID.i status.jobID.i .
  END
  /*
   *  complete the new job initialization into the server lists
   */
```

```
  module.jobID.0 = i - 1
  oleclip.jobID = oleclip. || '.' || jobID
  olepipe.jobID = olepipe. || '.' || jobID
  olewin.jobID = olewin. || '.' || jobID
  olewin.jobID.0 = ''
  userport.jobID = GETARG(pkt,1)
  userscreen.jobID = GETARG(pkt,2)
  CALL  IniModule (jobID,1)
END
```

1.29 newprefs

```
/*
 *  NEWPREFS jobID modID
 *
 */
WHEN cmd = 'NEWPREFS' THEN DO

  DO jobID = 0 TO module.0
    IF SHOW('P',olewin.jobID) THEN CALL CloseWindow(olewin.jobID,"CONTINUE")
  END

  CALL DELAY(delaytime * 50)
  CALL  ReadLocale (0,1)
  CALL  ReadConfig (0,1)
  CALL  GetServerConfig ()

  DO jobID = 0 TO module.0

    IF SHOW('P',olewin.jobID) & oleind.jobID = 0 THEN DO

      modID = olewin.jobID.0
      IF ~ NewWindow (jobID,modID, GetLocale (jobID,modID,'TITLE')) THEN
        CALL  FreeJob (jobID)
      ELSE
        IF jobID ~= 0 THEN INTERPRET 'ADDRESS' oleport.jobID.modID 'UNICONIFY'

      END

    oleind.jobID = 0
  END

END
```

1.30 about

```
/*
 *  ABOUT jobID modID strID
 *
 *  ERROR jobID modID code text
 *
 *  0)  general error
```

```
 *  1)  can't open message port %s
 *  2)  can't open IDCMP %s
 *  3)  unknown command %s
 *  4)  error writing file
 *  5)  error reading file
 */
WHEN cmd = 'ABOUT' | cmd = 'ERROR' THEN DO
  PARSE VAR argv a1 a2

  IF cmd = 'ABOUT' THEN DO
    a1 =  GetLocale (jobID,modID,a1,SUBSTR(ver_TAG,7))
    a2 = GetLocale(0,1,'OK1')
    END

  ELSE DO
    a1 = GetLocale(0,1,'ERR_' || a1,STRIP(a2,'B'))
    a2 = GetLocale(0,1,'OK2')
    END

  CALL RTezRequest(a1,a2,tit_TAG,rt_TAG 'rt_pubscrname=' || userscreen.jobID)
END
```

1.31 quit

```
/*
 *  QUIT
 *
 *  The "QUIT" command close all windows, exit from all modules
 *  and free all allocated by user jobs.
 */
WHEN cmd = 'QUIT' THEN
  IF RTezRequest( GetLocale (0,1,10),GetLocale(0,1,'OK5'),tit_TAG,rt_TAG) = 0 THEN
    cmd = ''
```

1.32 ole jobs

```
In the OLE System, a "job" is a set of one or more  modules .
It also include other information about user execution environment:

  the caller application
  its screen name

  module1
    .
    .
    .
  moduleN

All this datas are stored in  server lists .
A new job is created when a user execute a "special" ARexx macro,
that I call  OLE startup script .
```

1.33 ole modules

```
In the OLE System, a "module" is part of a  job .
Each one has own jobID and modID. These are numbers assigned
by the server every time a new job is created.
```