spUtils

Rexx Utility Package

Reference Manual

Preview Release 1

© 2004 by mecking.net

Table of Contents

 © 2004 by mecking.net

Introduction

1. Software License

If you use spUtils.dll or any other file from the archive you agree to these terms.

The files are licensed, not sold. You obtain no rights other than those granted to you under this license.

This license permits you to:

1. use the files on one or more machines, even at the same time.
2. make copies of the files.
3. redistribute the archive at no charge.

You may not sublicense, rent or lease the files.

There is no warranty that the files are free from claims by a third party of copyright, patent, trademark, trade secret, or any other intellectual property infringement.

The author or authors are under no circumstances liable for any of the following:

1. third-party claims against you for losses or damages.
2. loss of, or damage to, your records or data.
3. economic consequential damages (including lost profits or savings) or incidental damages, even if we are informed of their possibility.

There is no warranty that the provided informations are correct, or that the programs operate uninterrupted or error free.

If you do not agree to any of the above terms and conditions, or if any of the above terms and conditions are not legal due to local/country laws you are not granted a license and may not use the files.

IBM, OS/2, Presentation Manager, and Workplace Shell are trademarks or service marks of IBM Corporation.

Serenity Systems, eCS, and eComStation are trademarks of Serenity Systems International

Other company, product, and service names may be trademarks or service marks of others.

2. Requirements

required: spUtils should work on any OS/2 or eComStation system which supports the Open32 API.

useful: XRay by CodeSmith Software is very useful when writing scripts using the PM API functions. It shows informations about the window control under the mouse cursor.
You'll find it at hobbes: http://hobbes.nmsu.edu/cgi-bin/h-search?key=xray.

PMTree by IBM is even more powerful. It not only shows informations about window controls, but also allows to edit them.

It's also on hobbes: http://hobbes.nmsu.edu/cgi-bin/h-search?key=pmtree.

 © 2004 by mecking.net

3. Installation

Copy spUtils.dll into any directory listed in your libpath, e.g. \OS2\DLL on your boot volume.

4. Release Notes

This release is called "Preview Release 1". Several functions still need to be implemented. But it is already quite usable and most of the features are not available in other Libraries. For example Alessandro Cantatore wrote on his TABTRAY page, that he first wanted to write a REXX script for XCenter tray switching. With this library this would be possible. Or Innotek wrote in the readme of OS/2 Kit for Java, that REXX scripts cannot access the registry by default. Right, not by default, but by the use of this library's spRegistry function it is possible.

© 2004 by mecking.net

Function Reference

1. Base Functions

1.1. spLoadFuncs

Adds all spUtils functions.

Syntax: CALL spLoadFuncs

Example Code:

```
/* bootdrv.cmd (spUtils Example Code)                                    */
/* The env variable specified as argument is set to the drive letter of  */
/* the boot volume, or if called without parameter, writes it to stdout  */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

PARSE ARG varname rest

IF varname='' THEN SAY spGetBootdrive()
ELSE CALL VALUE varname,spGetBootdrive(),'OS2ENVIRONMENT'

CALL spDropFuncs
```

1.2. spDropFuncs

Removes all spUtils functions.

Syntax: CALL spDropFuncs

Example Code (part from example code of spLoadFuncs):

```
/* bootdrv.cmd (spUtils Example Code)                                    */
[...]
CALL spDropFuncs
```

 © 2004 by mecking.net

1.3. spVersion

Returns the version of the spUtils DLL used.

Syntax: version = spVersion ()

Returns: spUtils Version

Example Code:

```
/* spversion.cmd (spUtils Example Code)                                    */
/* Writes the version of the sputils.dll to stdout                         */

CALL RXFUNCADD 'spVersion','spUtils','spVersion'
SAY spVersion()
```

1.4. spSetAutoSleep

Sets the time interval to suspend in certain spUtils functions.

Syntax: CALL spSetAutoSleep time

Parameter: time – number of milliseconds to suspend

Example Code:

```
/* wininfo.cmd (spUtils Example Code)                                    */
/* waits for focus change and displays infos about the new focus window  */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 1000

hwnd=spQueryFocus()
SAY 'Change Focus to window control to be inspekted.'
do while hwnd=spQueryFocus()
     nop
end
SAY 'Focus change detected'
CALL spSetAutoSleep 50

hwnd=spQueryFocus()
CALL ShowInfo 'Focus window',hwnd
CALL ShowInfo 'Parent window',spQueryWindow(hwnd,'QW_PARENT')
CALL ShowInfo 'Owner window',spQueryWindow(hwnd,'QW_OWNER')
CALL ShowInfo 'Frame owner',spQueryWindow(hwnd,'QW_FRAMEOWNER')

RETURN

ShowInfo: PROCEDURE
     sp=LEFT('',20,' ')

     IF \spIsWindow(ARG(2)) THEN RETURN

     id=spQueryWindowID(ARG(2))
     IF spIsWindowEnabled(ARG(2)) THEN stat='enabled'; ELSE stat='disabled'
     IF spIsWindowShowing(ARG(2)) THEN showing='yes'; ELSE showing='no'
     IF spIsWindowVisible(ARG(2)) THEN visible='yes'; ELSE visible='no'

     SAY ''
     SAY LEFT(ARG(1),20,' ')||,
             'text:    "'||spQueryWindowText(ARG(2))||'"'
     SAY sp||'handle:  '||ARG(2)||' (0x'||spD2X(ARG(2),'1')||')'
     SAY sp||'id:      '||id||' (0x'||D2X(id)||')'
     SAY sp||'state:   '||stat
     SAY sp||'showing: '||showing
     SAY sp||'visible: '||visible

     RETURN
```


© 2004 by mecking.net

1.5. spD2X

This function performs a decimal to hexadecimal conversation. In contrast to D2X, spD2X can deal with ULONG values.

Syntax: hexVal = spD2X (decVal [, [type]])

Parameters: decVal – the value to be converted

type – type of value. If specified, leading zeros are returned. To get uppercase letters, you have to specify the type in uppercase.

type	*description*	*decimal value range*	*hexadecimal value range*
'n'	unsigned nibble (4 bit value)	0-15	0-F
'b'	unsigned byte (8 bit value)	0-255	00-FF
's'	unsigned short (2 byte value)	0-65536	0000-FFFF
'l'	unsigned long (4 byte value)	0-4294967295	00000000-FFFFFFFF

Table 1 spD2X value types

Returns: hexadecimal value

Example Code: (part of spSetAutoSleep example code)

```
/* wininfo.cmd (spUtils Example Code)                                    */
[...]
SAY sp||'handle:  '||ARG(2)||' (0x'||spD2X(ARG(2),'l')||')'
[...]
```

2. DOS API Functions

2.1. spQuerySysInfo

Returns values of static system variables.

Syntax: value = spQuerySysInfo (valueID)

Parameter: valueID – identifier of system variable to be queried.

valueID	*description*
QSV_MAX_PATH_LENGTH	Maximum length, in bytes, of a path name.
QSV_MAX_TEXT_SESSIONS	Maximum number of text sessions.
QSV_MAX_PM_SESSIONS	Maximum number of PM sessions.
QSV_MAX_VDM_SESSIONS	Maximum number of DOS sessions.
QSV_BOOT_DRIVE	Drive from which the system was started (1 means drive A, 2 means drive B, and so on).
QSV_DYN_PRI_VARIATION	Dynamic priority variation flag (0 means absolute priority, 1 means dynamic priority).
QSV_MAX_WAIT	Maximum wait in seconds.
QSV_MIN_SLICE	Minimum time slice in milliseconds.
QSV_MAX_SLICE	Maximum time slice in milliseconds.
QSV_PAGE_SIZE	Memory page size in bytes. This value is 4096 for the 80386 processor.
QSV_VERSION_MAJOR	Major version number.
QSV_VERSION_MINOR	Minor version number.
QSV_VERSION_REVISION	Revision number.
QSV_MS_COUNT	Value of a 32-bit, free-running millisecond counter. This value is zero when the system is started.
QSV_TIME_LOW	Low-order 32 bits of the time in seconds since January 1, 1970 at 0:00:00.
QSV_TIME_HIGH	High-order 32 bits of the time in seconds since January 1, 1970 at 0:00:00.
QSV_TOTPHYSMEM	Total number of bytes of physical memory in the system.
QSV_TOTRESMEM	Total number of bytes of resident memory in the system.
QSV_TOTAVAILMEM	Maximum number of bytes of memory that can be allocated by all processes in the system. This number is advisory and is not guaranteed, since system conditions change constantly.
QSV_MAXPRMEM	Maximum number of bytes of memory that this process can allocate in its private arena. This number is advisory and is not guaranteed, since system conditions change constantly.
QSV_MAXSHMEM	Maximum number of bytes of memory that a process can allocate in the shared arena. This number is advisory and is not guaranteed, since system conditions change constantly.
QSV_TIMER_INTERVAL	Timer interval in tenths of a millisecond.

 © 2004 by mecking.net

valueID	*description*
QSV_MAX_COMP_LENGTH	Maximum length, in bytes, of one component in a path name.
QSV_FOREGROUND_FS_SESSION	Session ID of the current foreground full-screen session. Note that this only applies to full-screen sessions. The Presentation Manager session (which displays Vio-windowed, PM, and windowed DOS Sessions) is full-screen session ID 1.
QSV_FOREGROUND_PROCESS	Process ID of the current foreground process.

Table 2 Value IDs for spQuerySysInfo

Returns: Value of system variable.

Example Code:

```
/* uptime.cmd (spUtils Example Code)                                    */
/* writes the time since last IPL to stdout                             */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

sec=spQuerySysInfo('QSV_MS_COUNT')%1000
SAY sec%86400      || ' day(s) '    ||,
    (sec%3600)//24 || ' hour(s) '   ||,
    (sec%60)//60   || ' minute(s) ' ||,
    sec//60        || ' second(s)'
```

2.2. spSleep

Suspends the current thread for a specified time interval.

Syntax: CALL spSleep msec

Parameter: msec – time to suspend in milliseconds

Example Code:

```
/* msleep.cmd (spUtils Example Code)                                    */
/* Waits the given number of milliseconds                               */

CALL RXFUNCADD 'spSleep','spUtils','spSleep'
PARSE ARG ms rest

CALL spSleep ms
```

© 2004 by mecking.net

2.3. spKillProcess

Flags a process to end, and returns the termination code to its parent (if any).

Syntax: success = spKillProcess ([[action] ,] pid)

Parameters: action – The value 'DKP_PROCESSTREE' is used to kill a process and all its descendant processes. The process must be either the current process, or it must have been directly created by the current process using DosExecPgm with a value of 2 (EXEC_ASYNCRESULT) for execFlag.

'DKP_PROCESS' is used to kill any process. This is the default value.

Aliases for 'DKP_PROCESSTREE': 'PROCESSTREE', 'TREE', 'T'.

Aliases for 'DKP_PROCESS': 'PROCESS', 'P'

pid – The process id of the process to be terminated.

Returns: 0 – No error.

13 – Error, invalid data.

217 – Error, zombie process.

303 – Error, invalid process id

305 – Error, not descendant

Example Code:

```
/* killpid.cmd (spUtils Example Code)                                    */
/* Kills the process with the pid passed as argument                     */

CALL RXFUNCADD 'spKillProcess','spUtils','spKillProcess'
PARSE ARG pid rest

IF spKillProcess(pid)\=0 THEN DO
    SAY 'Failed to kill pid ' || pid
END
ELSE DO
    SAY 'Pid ' || pid || ' has been killed.'
END
```

 © 2004 by mecking.net

3. Functions related to DOS API

3.1. spKillAll

The full-qualified file names of the modules running as processes on the system are matched against a substring. On success, the process gets flagged to end.

Syntax: numberKilled = spKillAll ([[action] ,] substring)

Parameters: action – The value 'DKP_PROCESSTREE' is used to kill a process and all its descendant processes. The process must be either the current process, or it must have been directly created by the current process using DosExecPgm with a value of 2 (EXEC_ASYNCRESULT) for execFlag.

'DKP_PROCESS' is used to kill any process. This is the default value.

Aliases for 'DKP_PROCESSTREE': 'PROCESSTREE', 'TREE', 'T'.

Aliases for 'DKP_PROCESS': 'PROCESS', 'P'

substring – Part of the filename. Prepend a backslash to full filenames to prevent the kill of processes which filenames contain substring as suffix, e.g. XFILE.EXE and E.EXE.

Returns: Number of processes killed.

Example Code:

```
/* killall.cmd (spUtils Example Code)                                    */
/* Kills all processes which run the specified exe file.                 */

CALL RXFUNCADD 'spKillAll','spUtils','spKillAll    '
PARSE UPPER ARG exe rest
IF POS('\',exe)=0 THEN exe='\' || exe
IF RIGHT(exe,4)\='.EXE' THEN exe=exe || '.EXE'

SAY spKillAll(exe) || ' process(es) killed'
```

© 2004 by mecking.net

3.2. spGetPidList

Retrieves a list of the process id's of all processes which module file names (full qualified) match against the given substring.

Syntax: success = spGetPidList (stem [, filter])

Parameters: stem – The variable stem into which the list get stored.

filter – Substring of the filename of those processes which should be found. Prepend a backslash to full filenames to prevent those processes which filenames contain filter as suffix from being found, e.g. XFILE.EXE and E.EXE.

Returns: True (1) – At least one process identifier has been returned

False (0) – No process identifier has been returned

Example Code: (part of example code of spFilename from

```
/* ps.cmd (spUtils Example Code)                                          */
/* List all currently active processes with pid and module name,          */
/* optionally a filter substring can be passed as argument.               */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

CALL spGetPidList 'p',ARG(1)
DO i=1 to p.0
     SAY RIGHT(p.i,5,' ') || ' ' || spFilenameFromPid(p.i)
END
```

3.3. spFilenameFromPid

Retrieves the full-qualified file name of the module running in the process with the specified process id.

Syntax: filename = spFilenameFromPid (pid)

Parameter: pid – process id of the process which module filename is to be retrieved.

Returns: filename of process module.

Example Code: (part of spGetPidList example code)

```
/* ps.cmd (spUtils Example Code)                                          */
[...]
SAY RIGHT(p.i,5,' ') || ' ' || spFilenameFromPid(p.i)
[...]
```

 © 2004 by mecking.net

3.4. spGetBootdrive

Returns the drive letter of the volume the system was booted from. In contrast to spQuerySysInfo('QSV_BOOT_DRIVE') a letter with appended colon is returned.

The function is equivalent to the following Rexx Statement:

```
D2C(C2D('A')-1+spQuerySysInfo('QSV_BOOT_DRIVE'))||':'
```

Syntax: bootdrive = spGetBootdrive ( )

Returns: Drive letter of boot volume followed by a colon

Example Code:

```
/* bootdrv.cmd (spUtils Example Code)                                    */
/* The env variable specified as argument is set to the drive letter of  */
/* the boot volume, or if called without parameter, writes it to stdout  */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

PARSE ARG varname rest

IF varname='' THEN SAY spGetBootdrive()
ELSE CALL VALUE varname,spGetBootdrive(),'OS2ENVIRONMENT'

CALL spDropFuncs
```

© 2004 by mecking.net

4. PM API WinDialogs Functions

4.1. spQueryDlgItemText

This function queries a text string in a dialog item.

Syntax: text = spQueryDlgItemText (hwnd , id)

Parameters: hwnd – The parent window handle. (Handle of the dialog owning the item to be queried.

id – Identity of the child window whose text is to be queried.

Returns: The text string that is obtained from the dialog item.

Example Code:

```
/* findinf1.cmd (spUtils Example Code)                                    */
/* Starts PMSEEK to find all *.INF files on the boot drive containing    */
/* the String passed as argument and writes a summery of the results to  */
/* stdout.                                                                */
CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
PARSE ARG searchstr
'START PMSEEK '||spGetBootdrive()||'*.INF '||searchstr
CALL spSleep 1000
hwnd=spFindWindowHandle(,,X2D('FA'),,,'PMSEEK.EXE')
IF hwnd=0 THEN DO
     SAY 'PMSeek window not found'
     RETURN
END
DO FOREVER
     text=spQueryDlgItemText(hwnd,X2D('3C'))
     IF LEFT(text,15)='Search Complete' THEN LEAVE
END
SAY text
SAY LEFT('',spQueryDlgItemTextLength(hwnd,X2D('3C')),'-')
```

4.2. spQueryDlgItemTextLength

Queries the length of a text string in a dialog item.

Syntax: len = spQueryDlgItemTextLength (hwnd , id)

Parameters: hwnd – The parent window handle. (Handle of the dialog owning the item to be queried.

id – Identity of the child window whose text length is to be queried.

Returns: Length of text.

Example Code: (part of spQueryDlgItemText example code)

```
/* findinf1.cmd (spUtils Example Code)                                    */
[...]
SAY LEFT('',spQueryDlgItemTextLength(hwnd,X2D('3C')),'-')
```

 © 2004 by mecking.net

4.3. spSetDlgItemText

This function sets a text string in a dialog item. Afterwards the text string of the dialog item is queried and returned.

Syntax: txt = spSetDlgItemText (hwnd , id , text)

Parameters: hwnd – The parent window handle. (Handle of the dialog owning the item to be queried.

id – Identity of the child window whose text length is to be queried.

text – The text to be set.

Returns: True (1) - Successful completion.

False (0) - Error occurred.

Example Code:

```
/* findinf.cmd (spUtils Example Code)                                    */
/* Starts PMSEEK to find all *.INF files on the boot drive containing    */
/* the String passed as argument and sets VIEW.EXE as editor.            */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

PARSE ARG searchstr
filemask=spGetBootdrive()||'*.INF'

CALL spSetAutoSleep 1000

'START PMSEEK 'filemask' 'searchstr

DO i=1 TO 2
     IF \spFindWindowHandles('hwnd',,,X2D('FA'),,,) THEN ITERATE
     DO j=1 TO hwnd.0
           hwndInput=spWindowFromId(hwnd.j,X2D('64'))
           IF hwndInput=0 THEN ITERATE
           IF spQueryWindowText(hwndInput)=filemask THEN DO
                 CALL spSetDlgItemText hwnd.j,X2D('6E'),'VIEW.EXE'
                 RETURN
           END
     END
END
SAY 'PMSEEK window not found'
```

5. PM API WinInput Functions

5.1. spQueryCapture

This function returns the handle of the window that has the pointer captured.

Syntax: hwnd = spQueryCapture ([hwndDesktop])

Parameter: hwndDesktop – Desktop-window handle. If ommited the default desktop handle (HWND_DESKTOP) will be used

Returns: 0 – No window has the pointer captured, or an error occurred.

>0 – Handle of the window with the pointer captured.

5.2. spQueryFocus

This function returns the focus window. It is NULLHANDLE if there is no focus window.

Syntax: hwnd = spQueryFocus ([hwndDesktop])

Parameter: hwndDesktop – Desktop-window handle. If ommited the default desktop handle (HWND_DESKTOP) will be used.

Returns: 0 – Error occurred or no focus window.

>0 – Handle of the focus window.

Example Code:

```
/* scclock.cmd (spUtils Example Code)                                    */
/* Switches the Smartcenter (aka WarpCenter or eComCenter) clock by      */
/* simulating a button click on the clock                                */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 0
focus=spQueryFocus()
PARSE VALUE spQueryPointerPos() WITH xptrpos yptrpos

hwnd=spFindWindowHandle(,,,'SmartCenter',,'\PMSHELL.EXE')
IF hwnd=0 THEN DO
     SAY 'Smartcenter (WarpCenter or eComCenter) not found'
     RETURN
END

xoffset=spQueryWindowPos(hwnd,'x')
yoffset=spQueryWindowPos(hwnd,'y')
xpos=spQueryWindowPos(hwnd,'cx')-5
ypos=5

CALL spSetPointerPos xoffset+xpos,yoffset+ypos
CALL spPostMsg hwnd,'WM_BUTTON1DOWN',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg hwnd,'WM_BUTTON1UP',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg focus,'WM_BUTTON1DOWN',spMPFROM2SHORT(0,0)
CALL spPostMsg focus,'WM_BUTTON1UP',spMPFROM2SHORT(0,0)
CALL spSetPointerPos xptrpos,yptrpos
```

 © 2004 by mecking.net

5.3. spSetFocus

This function sets the focus window.

Syntax: success = spSetFocus ([[hwndDesktop] ,] hwnd)

Parameter: hwndDesktop – Desktop window handle. If omitted the default desktop handle (HWND_DESKTOP) will be used.

hwnd – Window handle to receive the focus.

Returns: True (1) – Successful completion.

False (0) – Error occurred.

© 2004 by mecking.net

6. PM API WinMessageMgr Functions

6.1. spPostMsg

This function posts a message to the message queue associated with the window defined by hwnd.

Syntax: success = spPostMsg (hwnd , msgid [, mparam1 [, mparam2]])

Parameters: hwnd – Window handle or 0. If 0 then the message is posted into the queue associated with the current thread.

msgid – Message identity.

mparam1 – Parameter 1.

mparam2 – Parameter 2.

Returns: True (1) – Message successfully posted

False (0) – Message could not be posted; for example, because the message queue was full.

Example Code:

```
/* scclock.cmd (spUtils Example Code)                                    */
/* Switches the Smartcenter (aka WarpCenter or eComCenter) clock by      */
/* simulating a button click on the clock                                */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 0
focus=spQueryFocus()
PARSE VALUE spQueryPointerPos() WITH xptrpos yptrpos

hwnd=spFindWindowHandle(,,,'SmartCenter',,'\PMSHELL.EXE')
IF hwnd=0 THEN DO
     SAY 'Smartcenter (WarpCenter or eComCenter) not found'
     RETURN
END

xoffset=spQueryWindowPos(hwnd,'x')
yoffset=spQueryWindowPos(hwnd,'y')
xpos=spQueryWindowPos(hwnd,'cx')-5
ypos=5

CALL spSetPointerPos xoffset+xpos,yoffset+ypos
CALL spPostMsg hwnd,'WM_BUTTON1DOWN',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg hwnd,'WM_BUTTON1UP',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg focus,'WM_BUTTON1DOWN',spMPFROM2SHORT(0,0)
CALL spPostMsg focus,'WM_BUTTON1UP',spMPFROM2SHORT(0,0)
CALL spSetPointerPos xptrpos,yptrpos
```

 © 2004 by mecking.net

7. PM API WinPointers Functions

7.1. spQueryPointerPos

This function queries the pointer position.

Syntax: position = spQueryPointerPos ([[hwndDesktop ,] positionCode])

Parameters: hwndDesktop – Desktop-window handle. If ommited the default desktop handle (HWND_DESKTOP) will be used.

positionCode – Code of position value to be retrieved.

positionCode	***description***
x	Query horizontal (x) position of the Pointer.
y	Query vertical (y) position of the Pointer.
b	Query horizontal (x) and vertical (y) position of the Pointer. They will be returned space separated.

Table 3 Position codes for spQueryPointerPos.

Returns: "" – Error occured.

other – Pointer position.

Example Code:

```
/* scclock.cmd (spUtils Example Code)                                    */
/* Switches the Smartcenter (aka WarpCenter or eComCenter) clock by      */
/* simulating a button click on the clock                                */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 0
focus=spQueryFocus()
PARSE VALUE spQueryPointerPos() WITH xptrpos yptrpos

hwnd=spFindWindowHandle(,,,'SmartCenter',,'\PMSHELL.EXE')
IF hwnd=0 THEN DO
     SAY 'Smartcenter (WarpCenter or eComCenter) not found'
     RETURN
END

xoffset=spQueryWindowPos(hwnd,'x')
yoffset=spQueryWindowPos(hwnd,'y')
xpos=spQueryWindowPos(hwnd,'cx')-5
ypos=5

CALL spSetPointerPos xoffset+xpos,yoffset+ypos
CALL spPostMsg hwnd,'WM_BUTTON1DOWN',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg hwnd,'WM_BUTTON1UP',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg focus,'WM_BUTTON1DOWN',spMPFROM2SHORT(0,0)
CALL spPostMsg focus,'WM_BUTTON1UP',spMPFROM2SHORT(0,0)
CALL spSetPointerPos xptrpos,yptrpos
```

© 2004 by mecking.net

7.2. spSetPointerPos

This function sets the pointer position.

Syntax: success = spSetPointerPos ([hwndDesktop ,] PosX , PosY)

Parameters: hwndDesktop – Desktop-window handle. If ommited the default desktop handle (HWND_DESKTOP) will be used.

PosX – X-position of pointer in screen coordinates.

PosY – Y-position of pointer in screen coordinates.

Returns: True (1) – Pointer position successfully updated.

False (0) – Pointer position not successfully updated.

Example Code: (part of spQueryPointerPos example code)

```
/* scclock.cmd (spUtils Example Code)                                    */
[...]
CALL spSetPointerPos xptrpos,yptrpos
[...]
```

 © 2004 by mecking.net

8. PM API WinSys Functions

8.1. spQuerySysValue

Queries a system value.

Syntax: value = spQuerySysValue ([hwndDesktop ,] valueID)

Parameters: hwndDesktop – Desktop-window handle. If ommited the default desktop handle (HWND_DESKTOP) will be used.

valueID – Identifier of system value to be queried.

valueID	*Description*
SV_ALARM	0 if the alarm sound generated by WinAlarm is disabled.
SV_ALTMNEMONIC	0 if the mnemonic is made up of ROMAN characters, other values if the mnemonic is made up of KATAKANA characters.
SV_ANIMATION	0 when animation is set off.
SV_BEGINDRAG	Mouse begin drag (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_BEGINDRAGKB	Keyboard begin drag (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_BEGINSELECT	Mouse begin swipe select (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_BEGINSELECTKB	Keyboard begin swipe select (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_CICONTEXTLINES	Maximum number of lines that the icon text may occupy for a minimized window.
SV_CONTEXTHELP	Mouse control for pop-up menu (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_CONTEXTHELPKB	Keyboard control for pop-up menu (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_CONTEXTMENU	Mouse request pop-up menu (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_CONTEXTMENUKB	Keyboard request pop-up menu (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_CMOUSEBUTTONS	The number of buttons on the pointing device (zero if no pointing device is installed).
SV_CTIMERS	Count of available timers.
SV_CURSORLEVEL	The cursor hide level.
SV_CURSORRATE	Cursor blink rate, in milliseconds.
SV_CXBORDER	Width of the nominal-width border.
SV_CXBYTEALIGN	Horizontal count of pels for alignment.
SV_CXDBLCLK	Width of the pointing device double-click sensitive area. The default is the system-font character width.
SV_CXDLGFRAME	Width of the dialog-frame border.
SV_CXFULLSCREEN	Width of the client area when the window is full screen.

© 2004 by mecking.net

valueID	*Description*
SV_CXHSCROLLARROW	Width of the horizontal scroll-bar arrow bit maps.
SV_CXHSLIDER	Width of the horizontal scroll-bar thumb.
SV_CXICON	Icon width.
SV_CXICONTEXTWIDTH	Maximum number of characters per line allowed in the icon text for a minimized window.
SV_CXMINMAXBUTTON	Width of the minimize/maximize buttons.
SV_CXMOTIONSTART	The number of pels that a pointing device must be moved in the horizontal direction, while the button is depressed, before a WM_BUTTONxMOTIONSTR message is sent.
SV_CXPOINTER	Pointer width.
SV_CXSCREEN	Width of the screen.
SV_CXSIZEBORDER	Width of the sizing border.
SV_CXVSCROLL	Width of the vertical scroll-bar.
SV_CYBORDER	Height of the nominal-width border.
SV_CYBYTEALIGN	Vertical count of pels for alignment.
SV_CYDBLCLK	Height of the pointing device double-click sensitive area. The default is half the height of the system font character height.
SV_CYDLGFRAME	Height of the dialog-frame border.
SV_CYFULLSCREEN	Height of the client area when the window is full screen (excluding menu height).
SV_CYHSCROLL	Height of the horizontal scroll-bar.
SV_CYICON	Icon height.
SV_CYMENU	Height of the single-line menu height.
SV_CYMINMAXBUTTON	Height of the minimize/maximize buttons.
SV_CYMOTIONSTART	The number of pels that a pointing device must be moved in the vertical direction, while the button is depressed, before a WM_BUTTONxMOTIONSTR message is sent.
SV_CYPOINTER	Pointer height.
SV_CYSCREEN	Height of the screen.
SV_CYSIZEBORDER	Height of the sizing border.
SV_CYTITLEBAR	Height of the caption.
SV_CYVSCROLLARROW	Height of the vertical scroll-bar arrow bit maps.
SV_CYVSLIDER	Height of the vertical scroll-bar thumb.
SV_DBLCLKTIME	Pointing device double-click time, in milliseconds.
SV_DEBUG	0 indicates this is not a debug system.
SV_ENDDRAG	Mouse end drag (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_ENDDRAGKB	Keyboard end drag (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_ENDSELECT	Mouse select or end swipe select (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).

© 2004 by mecking.net

valueID	*Description*
SV_ENDSELECTKB	Keybaord select or end swipe select (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_ERRORDURATION	Duration for error alarms generated by WinAlarm.
SV_ERRORFREQ	Frequency for error alarms generated by WinAlarm.
SV_EXTRAKEYBEEP	When non 0, the press of a key that does not exist on the Enhanced keyboard causes the system to generate a beep.
SV_FIRSTSCROLLRATE	The delay (in milliseconds) before autoscrolling starts, when using a scroll bar.
SV_INSERTMODE	Non 0 if the system is in insert mode (for edit and multi-line edit controls); 0 if in overtype mode. This system value is toggled by the system when the insert key is toggled, regardless of which window has the focus at the time.
SV_KBDALTERED	Hardware ID of the newly attached keyboard. Note: The OS/2 National Language Support is only loaded once per system IPL. The OS/2 NLS translation is based partially on the type of keyboard device attached to the system. There are two main keyboard device types: PC AT styled and Enhanced styled. Hot Plugging between these two types of devices may result in typing anomalies due to a mismatch in the NLS device tables loaded and that of the attached device. It is strongly recommended that keyboard hot plugging be limited to the device type that the system was IPL'd with. In addition, OS/2 support will default to the 101/102 key Enhanced keyboard if no keyboard or a NetServer Mode password was in use during system IPL. (See Category 4, IOCtls 77h and 7Ah for more information on keyboard devices and types.)
SV_LOCKSTARTINPUT	0 when the type ahead function is disabled; non 0 when the type ahead function is enabled
SV_MENUROLLDOWNDELAY	The delay in milliseconds before displaying a pull down referred to from a submenu item, when the button is already down as the pointer moves onto the submenu item.
SV_MENUROLLUPDELAY	The delay in milliseconds before hiding a pull down referred to from a submenu item, when the button is already down as the pointer moves off the submenu item.
SV_MONOICONS	When non 0 preference is given to black and white icons when selecting which icon resource definition to use on the screen. Black and white icons may have more clarity than color icons on LCD and Plasma display screens.
SV_MOUSEPRESENT	When non 0 a mouse pointing device is attached to the system.
SV_NOTEDURATION	Duration for note alarms generated by WinAlarm.
SV_NOTEFREQ	Frequency for note alarms generated by WinAlarm.
SV_OPEN	Mouse open (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_OPENKB	Keyboard open (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).

© 2004 by mecking.net

valueID	*Description*
SV_POINTERLEVEL	Pointer hide level. If the pointer level is zero, the pointer is visible. If it is greater than zero, the pointer is not visible. The WinShowPointer call is invoked to increment and decrement the SV_POINTERLEVEL, but its value cannot become negative.
SV_PRINTSCREEN	Non 0 when the Print Screen function is enabled; 0 when the Print Screen function is disabled.
SV_SCROLLRATE	The delay (in milliseconds) between scroll operations, when using a scroll bar.
SV_SETLIGHTS	When non 0, the appropriate light is set when the keyboard state table is set.
SV_SINGLESELECT	Mouse select (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_TASKLISTMOUSEACCESS	Determines whether the task list is displayed when mouse buttons 1 and 2 are pressed simultaneously, or when mouse button 2 is pressed by itself, or for no mouse gesture.
SV_TEXTEDIT	Mouse begin direct name edit (low word=mouse message id (WM_*), high word=keyboard control code (KC_*)).
SV_TEXTEDITKB	Keyboard begin direct name edit (low word=virtual key code (VK_*), high word=keyboard control code (KC_*)).
SV_TRACKRECTLEVEL	The hide level of the tracking rectangle (zero if visible, greater than zero if not).
SV_SWAPBUTTON	Non 0 if pointing device buttons are swapped. Normally, the pointing device buttons are set for right-handed use. Setting this value changes them for left-handed use. If non 0, WM_LBUTTON* messages are returned when the user presses the right button, and WM_RBUTTON* messages are returned when the left button is pressed. Modifying this value affects the entire system. Applications should not normally read or set this value; users update this value by means of the user interface shell to suit their requirements.
SV_WARNINGDURATION	Duration for warning alarms generated by WinAlarm.
SV_WARNINGFREQ	Frequency for warning alarms generated by WinAlarm.

Table 4 Value IDs for spQuerySysValue

Returns: 0 – Error occurred

>0 – System value. Dimensions are in pels and times are in milliseconds.

© 2004 by mecking.net

Example Code:

```
/* startlpx.cmd (spUtils Example Code)                                    */
/* start wps object <LAUNCHPAD_x> where x is current screen width         */

CALL RXFUNCADD 'SysSetObjectData','RexxUtil','SysSetObjectData'CALL
RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

obj='<LAUNCHPAD_'||spQuerySysValue('SV_CXSCREEN')||'>'
CALL SysSetObjectData obj,'OPEN=DEFAULT'
```

9. PM API WindowMgr Functions

9.1. spIsChild

This function indicates if a window is a descendant of another window.

Syntax: ischild = spIsChild (hwnd , hwndParent)

Parameters: hwnd – Window handle of child window.

hwndParent – Window handle of parent window.

Returns: True (1) – if child window is a descendant of the parent window, or is equal to it.

False (0) – if child window is not a descendant of the parent, or is an Object Window (even if hwndParent is specified as the desktop or HWND_DESKTOP), or an error occurred.

9.2. spIsControlEnabled

This function returns the state (enabled/disabled) of the specified item in the dialog template within a dialog box.

Syntax: enabled = spIsControlEnabled (hwndDlg , id)

Parameters: hwndDlg – Dialog window handle.

id - Identity of the specified item.

Returns: True (1) – Specified item is enabled.

False (0) – Specified item is disabled.

9.3. spIsMenuItemChecked

This function returns the state (checked/not checked) of the identified menu item.

Syntax: checked = spIsMenuItemChecked (hwndMenu , id)

Parameters: hwndMenu – Menu window handle.

id – Identity of the menu item.

Returns: True (1) – Menu item is checked.

False (0) – Menu item is not checked.

 © 2004 by mecking.net

9.4. spIsMenuItemEnabled

This function returns the state (enabled/disabled) of the specified menu item.

Syntax: enabled = spIsMenuItemEnabled (hwndMenu , id)

Parameters: hwndMenu – Menu window handle.
id – Identity of the menu item.

Returns: True (1) – Menu item is enabled.
False (0) – Menu item is disabled.

9.5. spIsMenuItemValid

This function indicates if the specified menu item is a valid choice.

Syntax: valid = spIsMenuItemValid (hwndMenu , id)

Parameters: hwndMenu – Menu window handle.
id – Identity of the menu item.

Returns: True (1) – Menu item is valid.
False (0) – Menu item is invalid.

© 2004 by mecking.net

9.6. spIsWindow

This function determines if a window handle is valid.

Syntax: iswindow = spIsWindow (hwnd)

Parameter: hwnd – Window handle.

Returns: True (1) – Window handle is valid.
False (0) – Window handle is invalid.

Example Code:

```
/* wininfo.cmd (spUtils Example Code)                                    */
/* waits for focus change and displays infos about the new focus window  */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 1000

hwnd=spQueryFocus()
SAY 'Change Focus to window control to be inspekted.'
do while hwnd=spQueryFocus()
     nop
end
SAY 'Focus change detected'
CALL spSetAutoSleep 50

hwnd=spQueryFocus()
CALL ShowInfo 'Focus window',hwnd
CALL ShowInfo 'Parent window',spQueryWindow(hwnd,'QW_PARENT')
CALL ShowInfo 'Owner window',spQueryWindow(hwnd,'QW_OWNER')
CALL ShowInfo 'Frame owner',spQueryWindow(hwnd,'QW_FRAMEOWNER')

RETURN

ShowInfo: PROCEDURE
     sp=LEFT('',20,' ')

     IF \spIsWindow(ARG(2)) THEN RETURN

     id=spQueryWindowID(ARG(2))
     IF spIsWindowEnabled(ARG(2)) THEN stat='enabled'; ELSE stat='disabled'
     IF spIsWindowShowing(ARG(2)) THEN showing='yes'; ELSE showing='no'
     IF spIsWindowVisible(ARG(2)) THEN visible='yes'; ELSE visible='no'

     SAY ''
     SAY LEFT(ARG(1),20,' ')||,
             'text:    "'||spQueryWindowText(ARG(2))||'"'
     SAY sp||'handle:  '||ARG(2)||' (0x'||spD2X(ARG(2),'1')||')'
     SAY sp||'id:      '||id||' (0x'||D2X(id)||')'
     SAY sp||'state:   '||stat
     SAY sp||'showing: '||showing
     SAY sp||'visible: '||visible

     RETURN
```

 © 2004 by mecking.net

9.7. spIsWindowEnabled

This function returns the state (enabled/disabled) of a window.

Syntax: isenabled = spIsWindowEnabled (hwnd)

Parameter: hwnd – Window handle.

Returns: True (1) – Window is enabled.
False (0) – Window is disabled.

Example code: (part of spIsWindow example code)

```
/* wininfo.cmd (spUtils Example Code)                                    */
[...]
IF spIsWindowEnabled(ARG(2)) THEN stat='enabled'; ELSE stat='disabled'
[...]
```

9.8. spIsWindowShowing

This function determines whether any part of the window hwnd is physically visible.

Syntax: isshowing = spIsWindowShowing (hwnd)

Parameter: hwnd – Window handle.

Returns: True (1) – Some part of the window is displayed on the screen.
False (0) – The function returns FALSE in these situations:

- No part of the window is displayed on the screen
- The window is disabled
- The window update is disabled

Example Code: (part of spIsWindow example code)

```
/* wininfo.cmd (spUtils Example Code)                                    */
[...]
IF spIsWindowShowing(ARG(2)) THEN showing='yes'; ELSE showing='no'
[...]
```

9.9. spIsWindowVisible

This function returns the visibility state of a window.

Syntax: isvisible = spIsWindowVisible (hwnd)

Parameter: hwnd – Window Handle.

Returns: True (1) – Window and all its parents have the WS_VISIBLE style bit set on.
False (0) – Window or one of its parents have the WS_VISIBLE style bit set off.

Example Code: (part of spIsWindow example)

```
/* wininfo.cmd (spUtils Example Code)                                    */
[...]
IF spIsWindowVisible(ARG(2)) THEN visible='yes'; ELSE visible='no'
[...]
```

9.10. spQueryActiveWindow

This function returns the active window for HWND_DESKTOP, or other parent window.

Syntax: hwnd = spQueryActiveWindow ([hwndParent])

Parameter: hwndParent – Parent-window handle for which the active window is required. If ommited.

Returns: 0 – No window is active.
>0 – Active-window handle.

9.11. spQueryButtonCheckstate

Returns the checked state of the button control specified.

Syntax: isChecked = spQueryButtonCheckstate (hwndDlg , buttonID)

Parameters: hwndDlg – Dialog window handle which owns the button to be queried.
buttonID – Button control identity.

Returns: 0 – The button control is in unchecked state.
1 – The button control is in checked state.
2 – The button control is in indeterminate state.

9.12. spQueryLboxCount

Returns the number of items in the List Box.

Syntax: num = spQueryLboxCount (hwndLbox)

Parameter: hwndLbox – List box handle.

Returns: Number of items in the list box.

 © 2004 by mecking.net

9.13. spQueryLboxSelectedItem

Returns the index of the selected item in the List Box (for single selection only).

Syntax: item = spQueryLboxSelectedItem (hwndLbox)

Parameter: hwndLbox – List box handle.

Returns: Index of the selected item in the list box.

9.14. spQueryWindow

Returns the window handle of a window that has a specified relationship to a specified window.

Syntax: hwndRelated = spQueryWindow (hwnd , relation)

Parameters: hwnd – Handle of window to be queried.

relation – Type of window information to be returned.

relation code	*description*
QW_NEXT	Next window in z-order (window below).
QW_PREV	Previous window in z-order (window above).
QW_TOP	Topmost child window.
QW_BOTTOM	Bottommost child window.
QW_OWNER	Owner of a window.
QW_PARENT	Parent of a window.
QW_NEXTTOP	Next window of the owner window hierarchy subject to their z-ordering.
QW_PREVTOP	Previous main window.
QW_FRAMEOWNER	Owner of hwnd mormalized so that it shares the same parent as hwnd.

Table 5 Relation codes for spQueryWindow

Returns: Handle of window related to hwnd.

Example code: (part of spIsWindow example code)

```
/* wininfo.cmd (spUtils Example Code)                                    */
[...]
hwnd=spQueryFocus()
CALL ShowInfo 'Focus window',hwnd
CALL ShowInfo 'Parent window',spQueryWindow(hwnd,'QW_PARENT')
CALL ShowInfo 'Owner window',spQueryWindow(hwnd,'QW_OWNER')
CALL ShowInfo 'Frame owner',spQueryWindow(hwnd,'QW_FRAMEOWNER')
[...]
```

9.15. spQueryWindowPos

This function queries the window size and position of a visible window.

Syntax: pos = spQueryWindowPos ([hwnd ,] key)

Parameters: hwnd – Window handle. If ommited the default desktop handle (HWND_DESKTOP) will be used.

key – Selects the position value to be queried.

Key	***Description***
cy height	Window height.
cx width	Window width.
y left	Y coordinate of the window's origin.
x bottom	X coordinate of the window's origin
hwndInsertBehind	Window behind which this window is placed.
fl flags options	Options.

Table 6 Position Value IDs for spQueryWindowPos

Returns: Window position value.

© 2004 by mecking.net

Example Code:

```
/* scclock.cmd (spUtils Example Code)                                    */
/* Switches the Smartcenter (aka WarpCenter or eComCenter) clock by      */
/* simulating a button click on the clock                                */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 0
focus=spQueryFocus()
PARSE VALUE spQueryPointerPos() WITH xptrpos yptrpos

hwnd=spFindWindowHandle(,,,'SmartCenter',,'\PMSHELL.EXE')
IF hwnd=0 THEN DO
     SAY 'Smartcenter (WarpCenter or eComCenter) not found'
     RETURN
END

xoffset=spQueryWindowPos(hwnd,'x')
yoffset=spQueryWindowPos(hwnd,'y')
xpos=spQueryWindowPos(hwnd,'cx')-5
ypos=5

CALL spSetPointerPos xoffset+xpos,yoffset+ypos
CALL spPostMsg hwnd,'WM_BUTTON1DOWN',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg hwnd,'WM_BUTTON1UP',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg focus,'WM_BUTTON1DOWN',spMPFROM2SHORT(0,0)
CALL spPostMsg focus,'WM_BUTTON1UP',spMPFROM2SHORT(0,0)
CALL spSetPointerPos xptrpos,yptrpos
```

9.16. spQueryWindowText

This function copies window text into a buffer.

Syntax: text = spQueryWindowText (hwnd)

Parameter: hwnd – Window handle. If hwnd is a frame-window handle, the title-bar window text is copied.

Returns: Window text.

Example Code: (part of spIsWindow example code)

```
/* wininfo.cmd (spUtils Example Code)                                    */
[...]
SAY LEFT(ARG(1),20,' ')||,
        'text:    "'||spQueryWindowText(ARG(2))||'"'
[...]
```

9.17. spSetWindowText

This function sets the window text for a specified window.

Syntax: success = spSetWindowText (hwnd , text)

Parameters: hwnd – Window handle of window control which text has to be updated

text – The text to be set.

Returns: True (1) – Text updated.

False (0) – Error occurred.

9.18. spWindowFromID

This function returns the handle of the child window with the specified identity.

Syntax: hwnd = spWindowFromID (hwndParent , id)

Parameters: hwndParent – Parent-window handle.

id – Identity of the child window. It must be greater or equal to 0 and less or equal to 0xFFFF.

Returns: 0 – No child window of the specified identity exists.

>0 – Child-window handle.

Example Code:

```
/* findinf.cmd (spUtils Example Code)                                    */
/* Starts PMSEEK to find all *.INF files on the boot drive containing    */
/* the String passed as argument and sets VIEW.EXE as editor.            */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

PARSE ARG searchstr
filemask=spGetBootdrive()||'*.INF'

CALL spSetAutoSleep 1000

'START PMSEEK 'filemask' 'searchstr

DO i=1 TO 2
     IF \spFindWindowHandles('hwnd',,,X2D('FA'),,,) THEN ITERATE
     DO j=1 TO hwnd.0
           hwndInput=spWindowFromId(hwnd.j,X2D('64'))
           IF hwndInput=0 THEN ITERATE
           IF spQueryWindowText(hwndInput)=filemask THEN DO
                 CALL spSetDlgItemText hwnd.j,X2D('6E'),'VIEW.EXE'
                 RETURN
           END
     END
END
SAY 'PMSEEK window not found'
```


© 2004 by mecking.net

10. Functions related to PM API

10.1. spFindWindowHandle

This function returns the window handle of the first window found, which matches certain criteria.

Syntax: hwnd = spFindWindowHandle ([hwndParent] , [class] , [id] , [text] , [pid] , [filename])

Parameters: hwndParent – Window handle of the parent window. If ommited, the default desktop handle will be used (HWND_DESKTOP).

class – Window class name.

id – Window identifier.

text – Window text (title).

pid – Process identifier of window owner

filename – Substring of the full-qualified module filename of the process owning the window.

Returns: 0 – No window which matches the criteria has been found.

>0 – Window handle of the first window found, which matches the criteria.

Example Code:

```
/* scclock.cmd (spUtils Example Code)                                    */
/* Switches the Smartcenter (aka WarpCenter or eComCenter) clock by      */
/* simulating a button click on the clock                                */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 0
focus=spQueryFocus()
PARSE VALUE spQueryPointerPos() WITH xptrpos yptrpos

hwnd=spFindWindowHandle(,,,'SmartCenter',,'\PMSHELL.EXE')
IF hwnd=0 THEN DO
     SAY 'Smartcenter (WarpCenter or eComCenter) not found'
     RETURN
END

xoffset=spQueryWindowPos(hwnd,'x')
yoffset=spQueryWindowPos(hwnd,'y')
xpos=spQueryWindowPos(hwnd,'cx')-5
ypos=5

CALL spSetPointerPos xoffset+xpos,yoffset+ypos
CALL spPostMsg hwnd,'WM_BUTTON1DOWN',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg hwnd,'WM_BUTTON1UP',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg focus,'WM_BUTTON1DOWN',spMPFROM2SHORT(0,0)
CALL spPostMsg focus,'WM_BUTTON1UP',spMPFROM2SHORT(0,0)
CALL spSetPointerPos xptrpos,yptrpos
```

© 2004 by mecking.net

10.2. spFindWindowHandles

Searches for windows which match certain criteria and stores them in an stem variable.

Syntax: numberFound = spFindWindowHandles(stem , [hwndParent] , [class] , [id] , [text] , [pid] , [filename])

Parameters: stem – Stem variable

hwndParent – Window handle of the parent window. If ommited, the default desktop handle will be used (HWND_DESKTOP).

class – Window class name.

id – Window identifier.

text – Window text (title).

pid – Process identifier of window owner

filename – Substring of the full-qualified module filename of the process owning the window.

Returns: Number of windows found. (Same value as stem.0.)

Example Code:

```
/* findinf.cmd (spUtils Example Code)                                    */
/* Starts PMSEEK to find all *.INF files on the boot drive containing    */
/* the String passed as argument and sets VIEW.EXE as editor.            */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs

PARSE ARG searchstr
filemask=spGetBootdrive()||'*.INF'

CALL spSetAutoSleep 1000

'START PMSEEK 'filemask' 'searchstr

DO i=1 TO 2
     IF \spFindWindowHandles('hwnd',,,X2D('FA'),,,) THEN ITERATE
     DO j=1 TO hwnd.0
           hwndInput=spWindowFromId(hwnd.j,X2D('64'))
           IF hwndInput=0 THEN ITERATE
           IF spQueryWindowText(hwndInput)=filemask THEN DO
                 CALL spSetDlgItemText hwnd.j,X2D('6E'),'VIEW.EXE'
                 RETURN
           END
     END
END
SAY 'PMSEEK window not found'
```

 © 2004 by mecking.net

10.3. spMPFROM2SHORT

Returns a message parameter (used by spPostMsg) value which is build from 2 short (integer) values.

Syntax: mparam = spMPFROM2SHORT (short1 , short2)

Parameters: short1 - Numeric value of type SHORT (-32768<=short1<=32767) or USHORT (0<=short1<=65535)

short2 - Numeric value of type SHORT (-32768<=short1<=32767) or USHORT (0<=short1<=65535)

Returns: Message parameter value.

Example Code: (part of example code of spFindWindowHandles)

```
/* scclock.cmd (spUtils Example Code)                                    */
[...]
CALL spPostMsg hwnd,'WM_BUTTON1DOWN',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg hwnd,'WM_BUTTON1UP',spMPFROM2SHORT(xpos,ypos)
CALL spPostMsg focus,'WM_BUTTON1DOWN',spMPFROM2SHORT(0,0)
CALL spPostMsg focus,'WM_BUTTON1UP',spMPFROM2SHORT(0,0)
[...]
```

10.4. spMPFROMSH2CH

Returns a message parameter (used by spPostMsg) value which is build from one short (integer) values and two character (integer) values.

Syntax: mparam = spMPFROMSH2CH (short , char1 , char2)

Parameters: short – Numeric value of type SHORT (-32768<=short1<=32767) or USHORT (0<=short1<=65535)

char1 – Numeric value of type UCHAR (0<=char1<=255)

char2 – Numeric value of type UCHAR (0<=char1<=255)

Returns: Message parameter value.

© 2004 by mecking.net

10.5. spQueryWindowID

Returns the identifier of the window.

Syntax: id = spQueryWindowID (hwnd)

Parameter: hwnd – Window handle of window control to be queried.

Returns: Identifier of window control.

Example Code:

```
/* wininfo.cmd (spUtils Example Code)                                    */
/* waits for focus change and displays infos about the new focus window  */

CALL RXFUNCADD 'spLoadFuncs','spUtils','spLoadFuncs'
CALL spLoadFuncs
CALL spSetAutoSleep 1000

hwnd=spQueryFocus()
SAY 'Change Focus to window control to be inspekted.'
do while hwnd=spQueryFocus()
     nop
end
SAY 'Focus change detected'
CALL spSetAutoSleep 50

hwnd=spQueryFocus()
CALL ShowInfo 'Focus window',hwnd
CALL ShowInfo 'Parent window',spQueryWindow(hwnd,'QW_PARENT')
CALL ShowInfo 'Owner window',spQueryWindow(hwnd,'QW_OWNER')
CALL ShowInfo 'Frame owner',spQueryWindow(hwnd,'QW_FRAMEOWNER')

RETURN

ShowInfo: PROCEDURE
     sp=LEFT('',20,' ')

     IF \spIsWindow(ARG(2)) THEN RETURN

     id=spQueryWindowID(ARG(2))
     IF spIsWindowEnabled(ARG(2)) THEN stat='enabled'; ELSE stat='disabled'
     IF spIsWindowShowing(ARG(2)) THEN showing='yes'; ELSE showing='no'
     IF spIsWindowVisible(ARG(2)) THEN visible='yes'; ELSE visible='no'

     SAY ''
     SAY LEFT(ARG(1),20,' ')||,
             'text:    "'||spQueryWindowText(ARG(2))||'"'
     SAY sp||'handle:  '||ARG(2)||' (0x'||spD2X(ARG(2),'1')||')'
     SAY sp||'id:      '||id||' (0x'||D2X(id)||')'
     SAY sp||'state:   '||stat
     SAY sp||'showing: '||showing
     SAY sp||'visible: '||visible

     RETURN
```


© 2004 by mecking.net

11. WIN API Registry Functions

11.1. spRegistry

- Mode 1: Setting single value.

Syntax: result = spRegistry (key , name , val , [type])

- Mode 2: Setting default value of a certain key.

Syntax: result = spRegistry (key , ['DEFAULT:'] , val)

- Mode 3: Querying single value.

Syntax: result = spRegistry (key , name)

- Mode 4: Query default value of a certain key.

Syntax: result = spRegistry (key [, ['DEFAULT:']])

- Mode 5: Query type of a single value.

Syntax: result = spRegistry (key , ['TYPE:'] , name)

- Mode 6: Deleting a single value.

Syntax: result = spRegistry (key , name , 'DELETE:')

- Mode 7: Deleting default value of a certain key.

Syntax: result = spRegistry (key , ['DEFAULT:'] , 'DELETE:')

- Mode 8: Deleting an key and all associated subkeys and values.

Syntax: result = spRegistry (key , 'DELETE:')

- Mode 9: Querying names of all values associated with a certain key.

Syntax: result = spRegistry (key , 'VALUES:' , 'stem')

- Mode 10: Querying subkeys of a certain key.

Syntax: result = spRegistry (key , 'SUBKEYS:' , 'stem')

Parameters: key – Registry key.

name – Name of the value to be set, queried, or deleted.

val – Value to be set, queried, or deleted.

stem – The result of the enumerating operations will be set into this stem veriable.

Returns: For successful setting invocations, result will equal ''. (1,2)

For successful querying invocations, result will be given the value of the specified registry value. (3,4,5)

For successful deleting invocations, result will equal ''. (6,7,8)

For succesful enumerating operations, result will be number of items returned. (9,10)

Possible types returned by mode 5 are 'dword:' (2 byte values), 'hex:' (variable length binary data), '' (variable length strings).

© 2004 by mecking.net

The error string 'ERROR:' may be returned if an error occurs. Additionally the Variable 'RC' will be set to the return code of the failing API Call.

Return Codes of common failure situations:

3 File Not Found (Key not found)

1010 Bad Key

Remarks: When setting a value of a non existing key, the key will be created.

Example Code:

```
/* regdump.cmd (spUtils Example Code)                                    */
/* Prints out a registry key with all values and subvalues, like the     */
/* Registry Export function of Regedit                                   */

PARSE ARG args
IF RIGHT(args,1)='\' THEN args=LEFT(args,LENGTH(args)-1)

SAY 'REGEDIT4'
SAY ''
CALL RegDump args
RETURN

RegDump:   PROCEDURE
     SAY '['||ARG(1)||']'

     CALL KeyDump ARG(1)

     CALL spRegistry ARG(1),'SUBKEYS:','s'
     DO i=1 TO s.0
           CALL RegDump ARG(1)||'\'||s.i
     END

     RETURN

KeyDump:   PROCEDURE
     CALL spRegistry ARG(1),'VALUES:','v'
     DO i=1 to v.0

           CALL ValDump ARG(1), v.i

     END
     CALL LINEOUT ,''
     RETURN

ValDump:   PROCEDURE
     IF ARG(2)='' THEN DO
           val=spRegistry(ARG(1),'DEFAULT:')
           type=''
           out='@='
     END
     ELSE DO
           val=spRegistry(ARG(1),ARG(2))
           type=spRegistry(ARG(1),'TYPE:',ARG(2))
           out='"'
           DO i=1 TO LENGTH(ARG(2))
                 c=SUBSTR(ARG(2),i,1)
                 IF c='\' | c='"' THEN out=out||'\'
                 out=out||c
           END
```

 © 2004 by mecking.net

```
            out=out||'"='type
      END
      SELECT
            WHEN type='dword:' THEN CALL DWordDump  val
            WHEN type='hex:'   THEN CALL HexDump    val
            WHEN type=''       THEN CALL StringDump val
            OTHERWISE EXIT 1
      END
      RETURN

DWordDump: PROCEDURE EXPOSE out
      DO i=1 TO LENGTH(out)
            c=SUBSTR(out,i,1)
      CALL LINEOUT ,out||spD2X(ARG(1),'l')
      RETURN

HexDump:   PROCEDURE EXPOSE out
      DO i=1 TO LENGTH(ARG(1))
            IF i>1 & LENGTH(out)>3 THEN out=out||','
            out=out||spD2X(C2D(SUBSTR(ARG(1),i,1)),'b')

            IF LENGTH(out)>75 & i<=LENGTH(ARG(1)) THEN DO
                  CALL LINEOUT ,out||',\'
                  out='  '
            END
      END
      IF out\='' THEN CALL LINEOUT ,out
      RETURN

StringDump:      PROCEDURE EXPOSE out
      CALL CHAROUT ,out||'"'
      posx=LENGTH(out)+1
      DO i=1 TO LENGTH(ARG(1))
            IF i>1 & posx>76 & i<LENGTH(ARG(1)) THEN DO
                  CALL LINEOUT ,'"\'
                  CALL CHAROUT ,'"'
                  posx=1
            END
            c=SUBSTR(ARG(1),i,1)
            IF c='\' | c='"' THEN DO
                  CALL CHAROUT ,'\'
                  posx=posx+1
            END
            CALL CHAROUT ,c
            posx=posx+1
      END
      CALL LINEOUT ,'"'
      RETURN
```

Alphabetical List of all Functions

 © 2004 by mecking.net