EuroForth 2010

Securing a Windows 7 Public Access System Using Forth

S.N. Arhipov Mg.Sc.Eng. apx@micross.co.uk
N.J. Nelson B.Sc C.Eng. M.I.E.E. njn@micross.co.uk
Micross Automation Systems, Units 4-5, Great Western Court, Ross-on-Way,
Herefordshire HR9 7XP, Tel. +44 1989 768080, Fax. +44 1989 768163.

Abstract

Very often in industrial conditions the real time program system is used by one or more operators, who have computer qualification and experience only in using this program. Therefore, it is very important, that this special group of users is allowed to use only this program and not allowed to use all other programs in the computer. At the same time it is necessary to keep multi user environment and allow the administrator to use all system opportunities. Herewith, only the administrator can switch between desktops, it should be quick and should continue executing the programs. In this article the Forth programmatical technique of disabling some functional features of Windows 7 is described.

Introduction

TRACKNET is a universal software package for tracking and control of work in commercial laundries. The software consists of two parts. The first is an operator interface program which runs on a network of Personal Computers (PCs) under the Windows operating system. It is a Windows application and therefore it can be run simultaneously with all business software. The TRACKNET operator interface program works with authorised users, who, accordingly, can have two roles: as an operator and as MICROSS administrator. These roles are specified and registered in the TRACKNET program and while using this program an operator has some functional limitations and also has no opportunity to switch to another program application or to press Ctrl+Alt+Del keys, because this action is intercepted and forbidden by the TRACKNET program.

Nevertheless, in the Windows Vista and Windows 7 operating systems it is impossible to intercept Ctrl+Alt+Del. Experiments with keyboard hooking, using SetWindowsHookEx() function, and the WM_HOTKEY message trapping code injection into the main windows procedure did not give a positive result because of the hook procedure

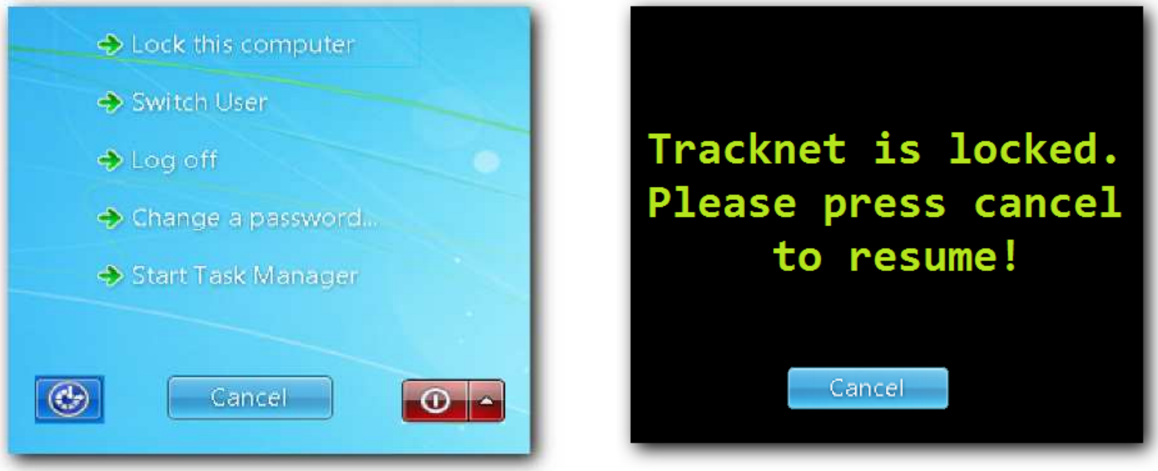
```
LRESULT KeyboardProc(...)
{
    if (Key == VK_CTRLALTDEL) return 1;
    return CallNextHookEx(...);
}
```

and hot key catching in Windows main procedure

```
LRESULT CALLBACK NewWindowProc
           (HWND  hWnd,  UINT  uMsg,   WPARAM  wParam,  LPARAM
lParam)
{
  if (uMsg == WM_HOTKEY)
     if (lparam == MAKELONG(MOD_CONTROL | MOD_ALT, VK_DELETE))
         return 1;
  return CallWindowProc(OldWindowProc, hWnd, wParam, lParam);
}
```

is activated later than `Ctrl+Alt+Del` typing event happens, an operating system sets focus on Windows log off screen.

Therefore, it was decided to use a desktop switching technique, creating a new desktop and run the `TRACKNET` program on it in order not to lock into program such system keys as `Alt+Esc`, `Alt+Tab`, `Ctrl+Shift+Esc` etc., but to isolate one process from another. Other processes continue running on the "`Default`" desktop and the screen saver runs on the "`Screen-saver`" desktop. All these desktops must be locked until the `TRACKNET` process runs on a new desktop and does not finish, or an authorised user switches back to the "`Default`" desktop.



Tracknet is locked. Please press cancel to resume!

Figure 1. Winlogoff desktop before and after commands disabling.

However, the logoff process runs on the "`Winlogoff`" desktop, which always switches on in Windows 7 and Windows Vista when the user presses `Ctrl+Alt+Del`. In this case, when creating a new desktop, it is necessary to change the background and disable all commands the "`Winlogoff`" desktop, except for pressing the “`Cancel`” button, and resetting the "`Winlogoff`" desktop to initial state, when an authorised user switches on the "`Default`" desktop. The initial state of the logoff screen and its transformation are illustrated in Figure 1.

Startup program overview

Program `TrStarter.exe` manages two processes: creating, if it does not exist, a new executable process `TRACKNET.EXE` in a new desktop, and disable all desktop commands: “`Log off`”, “`Lock this computer`”, “`Change a password...`”, “`Start Task Manager`”, “`Switch User`” as well as disable “`Shut down`” and “`Ease of access`” buttons (Figure 1.) on the "`Winlogoff`" desktop. On starting the program

TrStarter.exe a new desktop is switched on with a new background image backgroundDefault.jpg from a specially created directory Currentdirectory\DESKTOPRES. Now any user has an opportunity to work solely with TRACNET program.

All another programs continue running on the "Default" desktop. Only a user with corresponding privileges, using “File\Switch desktop” command, can switch the desktop back without closing the TRACNET program. Next time when a starter program tries to return the focus to a new desktop, it can recognize if the TRACNET program is running or not and correspondingly not create a new process. In the TRACNET program, with the help of the “File\Exit” command, the start up desktop takes the focus and TRACKNET is closed (Figure 2.).

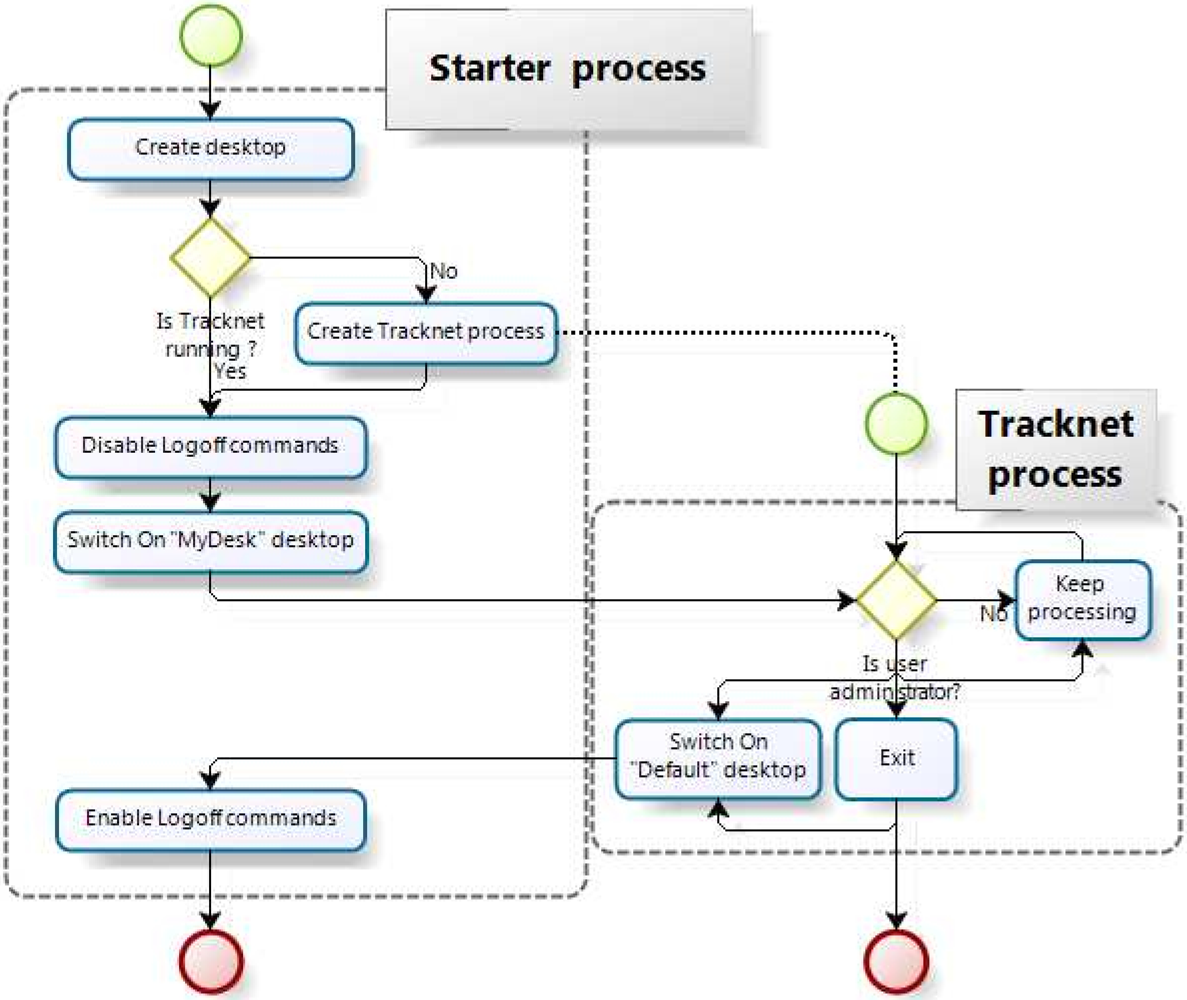


Figure 2. Starter TrStarter.exe and Tracknet.exe processes activity diagram.

TRACNET is a program which is implemented using ProForth for Windows V2.100 and which already has been successfully commercially used more than ten years [1]. TrStarter starter program is implemented using VFX Forth for Windows (4.41. 29 March 2010) [3] in order to execute the TRACKNET program in the operating systems Vista or Windows 7 [2] .It uses WinApi 32 functions for processes, windows registry, files and directories management.

C structures mapping in Forth

To create a new process it is necessary to use two structures. The first of them is `STARTUPINFO`, which specifies a window station, desktop, standard handles, and the appearance of the main window for a process at the time of creation. The table below specifies this structure in Forth and C notations, and fields, which are used in a process creation, are represented as comments in the third column of the table below.

Forth notation	*C definition*	*Used fields*
`STRUCT STARTUPINFO`	`typedef`	
	`struct  _STARTUPINFO {`	
`DWORD field SI.CB`	`DWORD  cb;`	\ Size of structure
`DWORD field lpReserved`	`LPTSTR lpReserved;`	
`DWORD field SI.LPDESKTOP`	`LPTSTR lpDesktop;`	\ Name of desktop
`DWORD field lpTitle`	`LPTSTR lpTitle;`	
`DWORD field dwX`	`DWORD  dwX;`	
`DWORD field dwY`	`DWORD  dwY;`	
`DWORD field dwXSize`	`DWORD  dwXSize;`	
`DWORD field dwYSize`	`DWORD  dwYSize;`	
`DWORD field dwXCountChars`	`DWORD  dwXCountChars;`	
`DWORD field dwYCountChars`	`DWORD  dwYCountChars;`	
`DWORD field dwFillAttribute`	`DWORD  dwFillAttribute;`	
`DWORD field dwFlags`	`DWORD  dwFlags;`	
`WORD field wShowWindow`	`WORD  wShowWindow;`	
`WORD field cbReserved2`	`WORD  cbReserved2;`	
`DWORD field lpReserved2`	`LPBYTE lpReserved2;`	
`DWORD field hStdInput`	`HANDLE hStdInput;`	
`DWORD field hStdOutput`	`HANDLE hStdOutput;`	
`DWORD field hStdError`	`HANDLE hStdError;`	
`END-STRUCT`	`}`	
	`STARTUPINFO,`	
	`*LPSTARTUPINFO;`	

The second structure that needed to create a new process is `PROCESSINFORMATION`. It contains information about the newly created process and its primary thread. In the table below this structure is specified in Forth and C notations. This structure is used as an output parameter of the `CreateProcess()` function in order to use process handle closing time.

Forth notation	*C language definition*
`STRUCT PROCESSINFORMATION`	`typedef struct _PROCESS_INFORMATION {`
`DWORD field hProcess`	`HANDLE hProcess;`
`DWORD field hThread`	`HANDLE hThread;`
`DWORD field dwProcessId`	`DWORD  dwProcessId;`
`DWORD field dwThreadId`	`DWORD  dwThreadId;`
`END-STRUCT`	`}`
	`PROCESS_INFORMATION,`
	`*LPPROCESS_INFORMATION;`

Entry function of starter program is

```
: RUN ( --- ) { | si[ STARTUPINFO ] pi[ PROCESSINFORMATION ]
res }
;
```

, where local variables `si[` and `pi[` are defined. Structure `si[` is initialised in a Forth program by setting values into two fields:

```
STARTUPINFO si[ SI.CB !                \ Set size of structure
```

```
DESKTOPNAME si[ SI.LPDESKTOP !      \ Set name of desktop
```

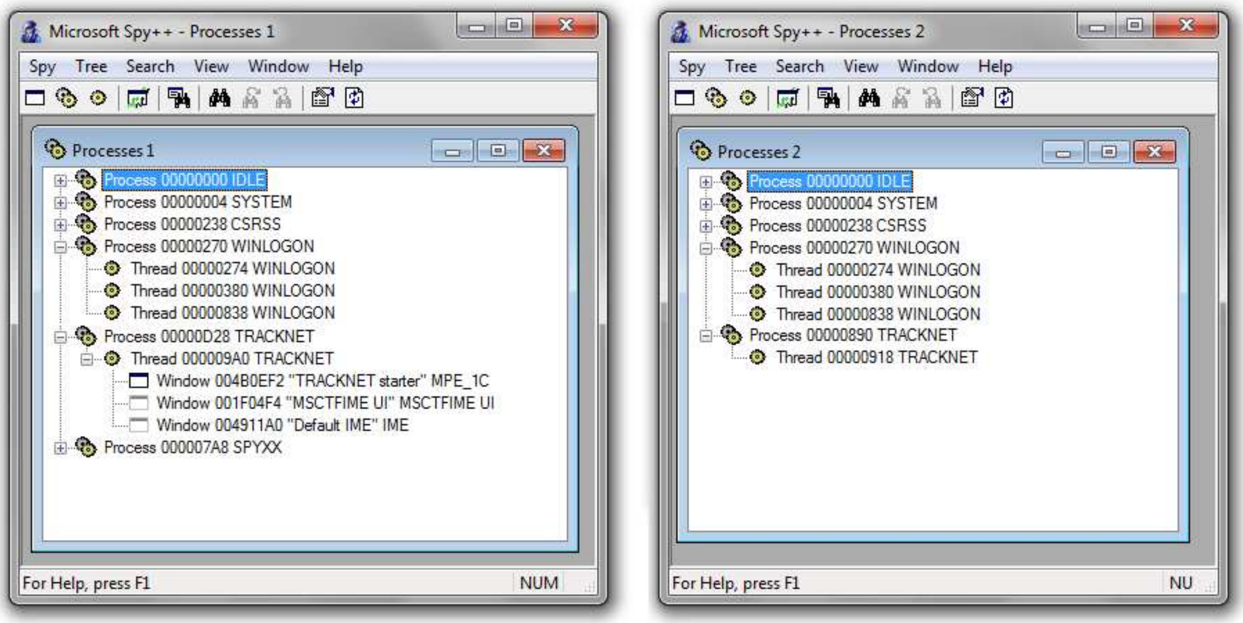
The function RUN creates a new desktop with the help of function CreateDesktop(), using parameter values in Forth notation:

Forth notation	*C notation*
\	HDESK CreateDesktop(
DESKTOPNAME \	in LPCTSTR lpszDesktop, \ The name of the desktop to be created.
NULL \	LPCTSTR lpszDevice, \ Reserved; must be NULL.
NULL \	DEVMODE * pDevmode, \ Reserved; must be NULL.
0 \	in DWORD dwFlags, \ The access to the desktop
DESKTOP_SWITCHDESKTOP \	in ACCESS_MASK dwDesiredAccess, \ The access to the desktop
NULL \	in LPSECURITY_ATTRIBUTES lpsa \ A pointer to a structure
CreateDesktop \	)

After a new desktop and startup information has been created TRACKNET process is created on a new desktop.

Process creation and monitoring

Using Win32-based Spy++ (SPYXX.EXE) utility, it is possible to view the system's processes, threads, windows, and window messages, but impossible to view the name of a desktop, or the name of both the desktop and window station for those processes. The starter program was executed in two different ways – on a default and a new separated desktop, and the results can be seen on Figure 3. When the starter program is running on a new desktop (right window), but Spy++ is running on a default desktop, it is possible to view only a process and its threads, but the actual desktop and windows of process are invisible. Anywhere Spy++ gives useful information about system's object names and its identification. It is important information, that can be used during testing of the process of creating new desktop.



```
Process 00000000 IDLE
Process 00000004 SYSTEM
Process 00000238 CSRSS
Process 00000270 WINLOGON
    Thread 00000274 WINLOGON
    Thread 00000380 WINLOGON
    Thread 00000838 WINLOGON
Process 00000D28 TRACKNET
    Thread 000009A0 TRACKNET
        Window 004B0EF2 "TRACKNET starter" MPE_1C
        Window 001F04F4 "MSCTFIME UI" MSCTFIME UI
        Window 004911A0 "Default IME" IME
Process 000007A8 SPYXX
```

```
Process 00000000 IDLE
Process 00000004 SYSTEM
Process 00000238 CSRSS
Process 00000270 WINLOGON
    Thread 00000274 WINLOGON
    Thread 00000380 WINLOGON
    Thread 00000838 WINLOGON
Process 00000890 TRACKNET
    Thread 00000918 TRACKNET
```

Figure 3. Starter TrStarter processes, threads and windows when running on "Default" desktop (left window) and when running on new desktop (right window).

The first step of the starter program `TrStarter` is to identify if the `TRACKNET` process is running or not. Forth function

```
: CHECKISTRACKNETRUNNING
   { | res processhandle modulehandle -- f }
;
```

examines all processes that are running (Figure 6.) and compares the names of the modules with the name `TRACKNET.EXE`. If there are no such names in the module name list then the function returns the value `false`, and for the rest, value `true`.

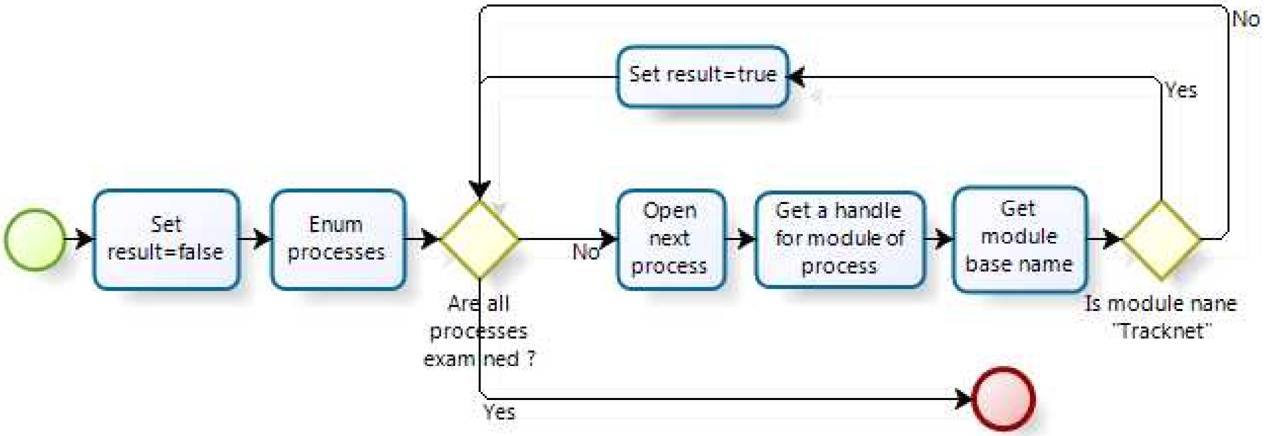


Figure 4. Activity diagram of function `CHECKISTRACKNETRUNNING`

There are three local variables in `CHECKISTRACKNETRUNNING` function. Result variable `res` is a temporary value storage. In the loop, variable `processhandle` sequentially sets the values, which are the handles of the all processes running in the system. Variable `modulehandle` sets a value, which is a handle of the module of each process. There may be many modules in one process, but function `CHECKISTRACKNETRUNNING` examines only first module in each process and therefore there is only one loop for looking over all processes.

The fist used WinApi function has the signature

```
BOOL WINAPI EnumProcesses
 (out DWORD *pProcessIds,in DWORD cb,out DWORD
*pBytesReturned);
```

and it retrieves the process identifier for each process object in the system. The use of this function is necessary in three objects:

- an array size constant, correspondent to input parameter `cb`

```
 4096 CONSTANT SIZEOFARR
```

- a pointer to an array of process identifiers, correspondent to the output parameter `pProcessIds`

```
 CREATE ARROFPROCIDS SIZEOFARR ALLOT
```

- the number of bytes returned in the array of process identifiers, correspondent to the output parameter `pBytesReturned`

```
 VARIABLE NUMBEROFBYTESRETURNED
```

In this case calling of `EnumProcesses()` function in Forth notation is:

```
ARROFPROCIDS SIZEOFARR NUMBEROFBYTESRETURNED EnumProcesses
DROP
```

When all system processes are enumerated it is possible to examine every process cyclically using the function

```
HANDLE OpenProcess ( in DWORD dwDesiredAccess,
                     in BOOL bInheritHandle,
                     in DWORD dwProcessId);
```

that opens an existing local process object and returnes the handle to the process according to its identifier. Function

```
BOOL   EnumProcessModules(in   HANDLE   hProcess,out   HMODULE
*lphModule, in DWORD cb, out LPDWORD lpcbNeeded);
```

retrieves a handle for each module in the specified process.

In every loop before process opening, into the variable, that is defined as

```
VARIABLE PROCESSID
```

the identifier of current process is loaded

```
ARROFPROCIDS I CELLS+ @ PROCESSID ! \ Set the current process ID
```

Using the current process identifier, which is the `PROCESSID` variable, calling the `OpenProcess()` function returns an open handle to the specified process.

```
PROCESS_QUERY_INFORMATION PROCESS_VM_READ OR \ The access to the process
FALSE                                        \ Processes do not inherit this handle
PROCESSID @                                  \ Current process ID
OpenProcess -> processhandle
```

Continuing a loop, the function `EnumProcessModules()` retrieves a list of handles of process modules

```
processhandle          \ A handle to the process from OpenProcess()
ADDR modulehandle      \ An array that receives the list of module handles
SIZEOFDWORD            \ The size of the array, in bytes\
NUMBEROFBYTESNEEDED    \ The number of bytes required to store handles in the array
EnumProcessModules -> res \ If the function succeeds, the return value is nonzero.
```

There are two specifics calling this function in the Forth program. The first of them is the use of the second parameter `modulehandle`, witch must be specified by type `HMODULE *`. Basically, it is an address of variable `modulehandle` and in Forth program the `ADDR` word is used. The second specific aspect is the using of the third parameter. In the function signature it is defined as `DWORD` type number, but it is not a count of elements in an array, but textually it is the size of the module handle address – simply constant

```
 4 CONSTANT SIZEOFDWORD
```

Last WinApi function in the loop is

```
DWORD   GetModuleBaseName(   in   HANDLE   hProcess,in   HMODULE
hModule,out LPTSTR lpBaseName,in DWORD nSize);
```

that retrieves the base name of the specified module. The handle of module is loaded in a local variable `modulehandle` and the handle of process is loaded in local variable `processhandle`

Therefore, calling this function

```
processhandle       \ A handle to the process that contains the module.
```

```
modulehandle         \ A handle to the module.
MODULEBASENAME       \ A pointer to the buffer that receives the base name of the module
MODULEBASENAMESIZE    \ The size of the buffer, in characters
GetModuleBaseName DROP
```

retrieves the base name of the specified module into the output parameter `MODULEBASENAME`. Its corresponding parameter in the function signature has `LPTSTR` type. In the Forth program this parameter is defined as a buffer with length `1024`:

```
 1024 CONSTANT MODULEBASENAMESIZE
 CREATE MODULEBASENAME MODULEBASENAMESIZE ALLOT
```

Likewise, the received value of the current module name is compared with the "`TRACKNET`" string and if they are equal then function `CHECKISTRACKNETRUNNING` returnes the value `true`.

Windows registry editing

When a new desktop has been created, it is necessary to disable all commands (Figure 1.) except "`Cancel`". This command disabling technique is based on Windows registry key editing. The data structure of Windows registry structure is represented in Figure 5. It is a hierarchical structure of keys and their sub keys. Every key can have many or no values at all; every key value is a structure with three attributes – name of value, type of value and data that is stored into the key value.

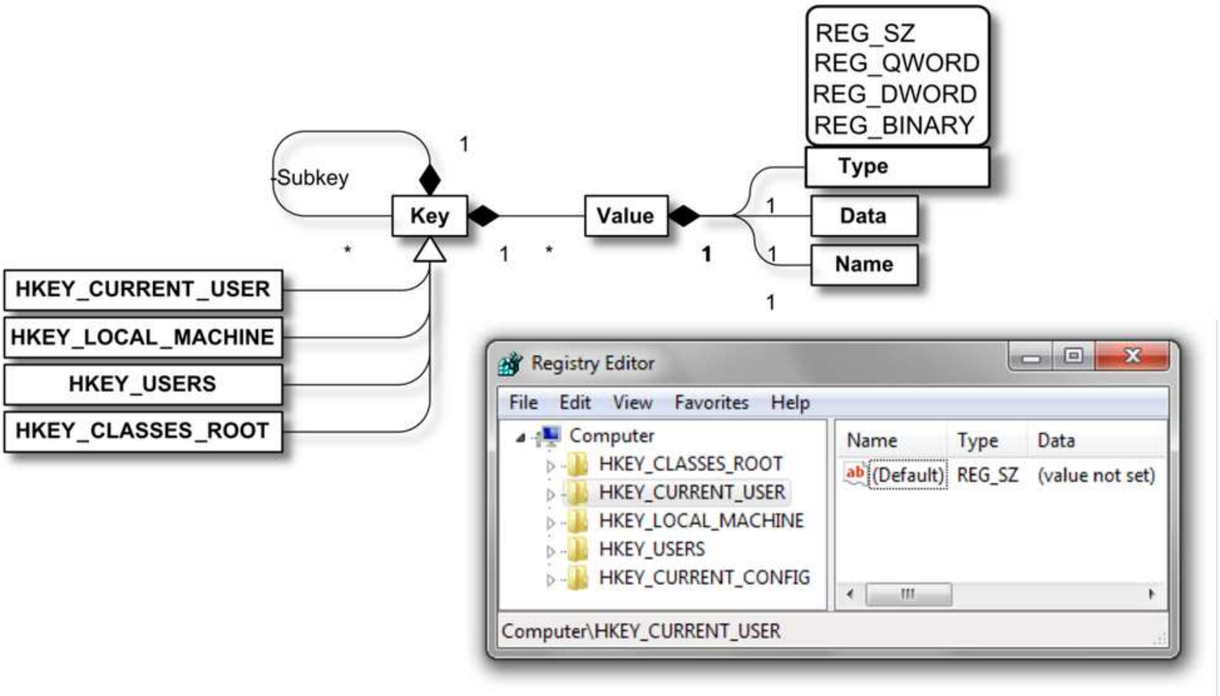


Figure 5. Structure of Windows registry

In order to disable a command on the logoff window, it is necessary in `HKEY_CURRENT_USER` root for `Software \Microsoft\Windows\CurrentVersion\Policies\Explorer` key to create a value with a corresponding name type and data as seen in table below.

Name of key value	*Type of key value*	*Data of key value*	*Disable action*
`NoLogoff`	`DWORD`	1	“Logg off”
`NoClose`	`DWORD`	1	“Shut Down”

But for the
`Software\Microsoft\Windows\CurrentVersion\Policies\System`
key it is necessary to create a value according to the table below.

Name of key value	*Type of key value*	*Data of key value*	*Disable action*
`DisableLockWorkstation`	`DWORD`	1	“Lock this computer”
`DisableChangePassword`	`DWORD`	1	“Change a password...”
`DisableTaskMgr`	`DWORD`	1	“Start Task Manager”
`HideFastUserSwitching`	`DWORD`	1	“Switch User”

In order to enable any command, the corresponding data of key value must be deleted or set to zero. It is possible to do it manually using the `regedit.exe` editor. But there are four WinApi functions for key management (Figure 6.) and two WinApi functions, with the help of which it is possible to manage values for the given key.

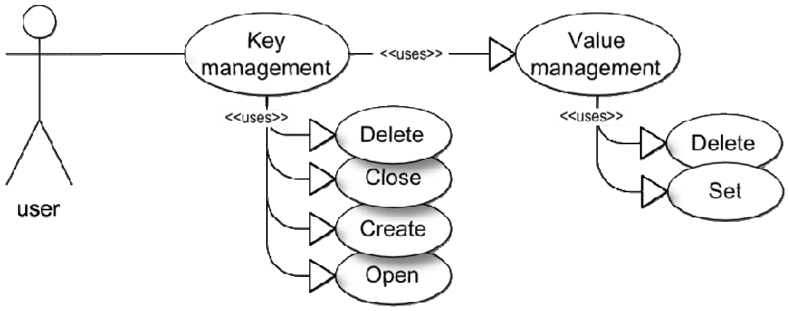


Figure 6. Functionality of Windows registry keys and values

The starter program uses such registry functions as:

- `RegOpenKeyEx()` - opens the specified registry key,
- `RegCloseKey()` - closes a handle to the specified registry key,
- `RegSetValueEx()` - sets the data and type of a specified value under a registry key,
- `RegDeleteKeyValue()` - removes the specified value from the specified registry key and sub key.

In the disable command the sequence of calling functions is shown on Figure 7. This is the activity sequence of the Forth word

```
: REGKEYSETVALUE ( predefinedkey keyname valuename value --- )
  { predefinedkey keyname valuename value | res }
;
```

, which set the value into the registry key. It receives input values into local variables

- `predefinedkey`, that must be a predefined constant `HKEY_CURRENT_USER`,
- `keyname`, that has one of two string values “`...\Explore`” or “`...\System`”,
- `valuename` - is one of the strings from the first column from the tables above,
- `value` – must be 1 to disable a command.

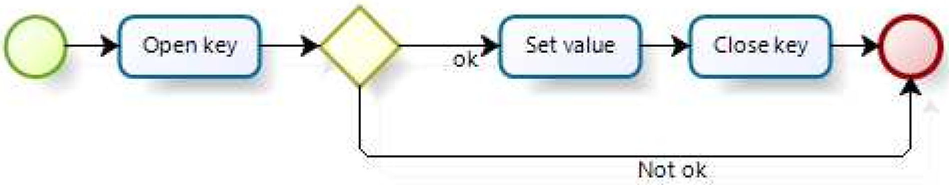


Figure 7. Activity diagram of function, that disable command

The fourth function `REGKEYSETVALUE` tries to open the key using function `RegOpenKeyEx()` with input parameter values `predefinedkey` and `keyname` (table below). If the function fails, the return value is a nonzero error code.

Forth notation *C language signature*

`\`	`LONG WINAPI RegOpenKeyEx(`
`predefinedkey\`	`in HKEY     hKey,` \ handle to registry key = HKEY_CURRENT_USER
`keyname      \`	`in LPCTSTR lpSubKey,` \ The name of the registry subkey to be opened.. f
`0            \`	`DWORD      ulOptions,` \ must be zero
`KEY_SET_VALUE\`	`in REGSAM  samDesired,`\ A mask that required to create, delete, or set a registry value
`PHKEYRESULT  \`	`out PHKEY  phkResult` \ A pointer to a variable that receives a handle to the opened key
`RegOpenKeyEx \`	`);`
`-> res`	

Key value setting is executed calling function `RegSetValueEx()` in Forth notation, that can be seen in table below.

Forth notation *C language signature*

	`LONG WINAPI RegSetValueEx(`
`PHKEYRESULT @       \`	`in        HKEY    hKey,` \ A handle to an open registry key
`valuename           \`	`in        LPCTSTR lpValueName,` \ The name of the value to be set
`0                   \`	`          DWORD   Reserved,` \ must be zero
`REG_DWORD           \`	`in        DWORD   dwType,` \ int type of data
`ADDR value          \`	`in const BYTE *  lpData,` \ The data to be stored
`4                   \`	`in        DWORD   cbData` \ The size of the data to be stored
`RegSetValueEx       \`	`);`

The use of `RegSetValueEx()` function has some particular features. The function fourth parameter `dwType` is predefined and has a value from the set – `REG_BINARY`, `REG_DWORD`, `REG_QWORD` etc. depending on the type (Figure 5.) of the value data to be stored. In the case of the command's disability it uses only value 1, that has type `int` or predefined value `REG_DWORD`. The second particular feature is that there are many of data types which it is possible to store in the registry key value. Therefore in the general case it is specified as `const BYTE *lpData` – pointer to first byte of memory segment with storage data. In Forth notation the `ADDR` word is used, which is the address of a variable value, that is received as an input parameter of the `REGKEYSETVALUE` word. The last parameter value of `RegSetValueEx()` function is simply `4` – the size of `int` type.

Function `RegCloseKey()` (table below) closes the opened registry key, using its handle as the parameter.

Forth notation	*C language signature*
PHKEYRESULT @ RegCloseKey	LONG WINAPI RegCloseKey(in HKEY hKey \ A handle to an open registry key);

Switching the "Default" desktop, program TRACKNET back on enables all logoff commands. It deletes key values NoLogoff, NoClose DisableLockWorkstation, DisableChangePassword, DisableTaskMgr, HideFastUserSwitching from the registry using function RegDeleteKeyValue (table below).

Forth notation	*C language signature*
predefinedkey keyname valuename RegDeleteKeyValue	LONG WINAPI RegDeleteKeyValue(in HKEY hKey, \ A handle to an open registry key. in LPCTSTR lpSubKey, \ The name of the registry subkey in LPCTSTR lpValueName \ The registry value to be removed from the key);

Desktop background changing

Using function CopyFile() a new background image, placed in a reserved folder and has the same name as original desktop image backgroundDefault.jpg, is copied into the folder C:\Windows\System32\oobe\Info\backgrounds and replaces the original background with the new one. Switching "Default" desktop back on, the original background image is restored from the reserved folder, replacing the image in its own folder. Function CopyFile() signature and its usage is represented in table below.

Forth notation

```
LOGINIMAGENAME    \ The name of an existing login image file from reserved folder.
LOGONINFOIMAGE    \ The name of the new login image file in C:\Windows\System32\oobe\Info\backgrounds
FALSE             \ Overwrite if file already exist
CopyFile
```

C language signature

```
BOOL CopyFile(
in  LPCTSTR lpExistingFileName,
in  LPCTSTR lpNewFileName,
in  BOOL    bFailIfExists);
```

In order to make the copying process successful, it is necessary to set corresponding permissions for the MICROSS administrator role. It is passible to do manually, using the Application Data Property editor for changing file ownership and security settings. But these steps can be executed programmatically.

Setting administrator permissions

To enable administrators to take ownership of shares is possible, using the command-line utility TakeOwn, but to manage security settings of files and folders is possible, using command-line tool Icacls. In order to use them, the Win32 function ShellExecute() is used in a program. This function signature and its usage are represented in the tables below.

Forth notation	*C language signature*
	`HINSTANCE ShellExecute(`
`0`	`in HWND hwnd,` \A handle to the owner window
`Z" open"`	`in LPCTSTR lpOperation,`\A verb, that specifies the action \ to be performed: „edit", „open", „print" etc.
`Z" cmd.exe"`	`in LPCTSTR lpFile,` \The object on which to execute \ the verb.
`Z" /c takeown /f C:\im.jpg"`	`in LPCTSTR lpParameters,`\ The parameters to be passed \ to the application
`Z" "`	`in LPCTSTR lpDirectory,` \ Pointer to working directory
`0`	`in INT nShowCmd` \The flag how to display \ application
`ShellExecute`	`);`

Forth notation

```
\ Applaing stored DACLs to files in specified directories for Administrators
  0
  Z" open"
  Z" cmd.exe"   \ open cmd.exe and execute Icacls utility
  Z"  /c Icacls C:\im.jpg /grant BUILTIN\Administrators:(F) "
  Z" "
  0
  ShellExecute
```

Conclusions

In result, the program `TrStart` allows to start any program in separated desktop and to restrict some users from all others program's execution, including task manager and all logoff commands. It is possible to tune the `TrStart` program in order to configure concrete user permissions. For users with the appropriate privilege it is enable to switch on default desktop by the new “`Switch desktop`” menu entry. The program `TrStart` allow return to the new desktop without restarting the base program, if it is already running.

Information sources

[1] N.J. Nelson, “Experiments in real time control in Windows using Forth”, 20th euroForth Conference, Dagstuhl Castle, November 19th-22th 2004.

[2] Yochay Kiriaty, Laurence Moroney, Sasha Goldshtein, Alon Fliess “Introducing Windows® 7 for Developers”, Microsoft Press, 2010.

[3] Microprocessor Engineering Limited, VFX Forth for Windows Native Code ANS Forth Implementation, User manual 4.41 29 March 2010