Process Info ActiveX Control for Microsoft® Windows™

Contents

1 Process Info ActiveX Control Overview **3**

1.1. Introduction *3*

1.2. Usage *3*

1.3. Interface Summary *3*

1.3.1. _DSKProcessInfo 3

1.4 Method Summary *4*

AboutBox 4

EnumProcesses 4

EnumApplications 4

EnumProcessModules 4

2. Methods **4**

2.1. AboutBox *4*

2.2. EnumProcesses *5*

2.3. EnumApplications *5*

2.4. EnumProcessModules *6*

2.5. EnumProcessThreads *7*

2.6. QuerySetPriorityClass *8*

2.7. QueryTerminateProcess *9*

3. Examples **9**

1 Process Info ActiveX Control Overview

1.1. Introduction

The Magneto Software Process Information ActiveX control (skprocessInfo.ocx) allows developers to monitor and control processes running on the current machine from their 32-Bit or 64-Bit applications.

It is a lightweight and powerful control that allows developers to retrieve some vital process information in real-time.

Process Information ActiveX control provides the following real-time vital processes information:

- List of running services, their names, process id's, details and status
- List of running processes, their names, process id's and status
- Process properties such as: process id, memory usage, base priority, full path name and threads information
- List of running applications and their status
- List of modules loaded by process
- Module properties such as full file name, base address, global usage, context usage and full description and version information.

The control can be used from any Windows-based applications development environment, including Visual Studio.
It comes with documentation, sample code, and working demo programs.

1.2. Usage

The control retrieves information about:

- Programs that are running
- Processes that are running
- Computer's Performance

1.3. Interface Summary

1.3.1. _DSKProcessInfo

Specifies a collection of logically grouped methods to retrieve processes/applications/modules related information.

AboutBox

EnumProcesses

EnumApplications

EnumProcessModules

EnumProcessThreads

QuerySetPriorityClass

QueryTerminateProcess

1.4 Method Summary

AboutBox
Display a dialog box with Skprocessinfo ActiveX control license and version information.

EnumProcesses
Retrieves list of processes that are running on the current machine

EnumApplications
Retrieves list of applications that are running on the current machine

EnumProcessModules
Retrieves list of modules that are used by the application. This list defines the set of files needed for the module to load and execute as a running process.

EnumProcessThreads

Retrieves list of threads executed by the process.

QuerySetPriorityClass

Sets the priority class for the specified process.

QueryTerminateProcess

Terminates the specified process and all of its threads.

2. Methods

2.1. AboutBox

Summary

Display a dialog box with SKProccessInfo control license and version information.

Syntax

void AboutBox();

Description

This method could be used to display version license information or to register skproccessInfo.ocx control.

Parameters

None.

Return value:

This function does not return a value.

2.2. EnumProcesses

Summary

Retrieves list of processes that are running on the current machine.

Syntax

void EnumProcesses (VARIANT* pvarProcesses);

Description

The process is assigned a process identifier. Until the process terminates, the process identifier uniquely identifies the process throughout the system.

Parameters

pvarProcessesis

[out] Pointer to a variant, containing two-dimensional SAFEARRAY. Each element of this SAFEARRAY is a VARIANT.

pvarProcesses receives a list of all running processes and their properties. The list describes the processes residing in the system address space when a snapshot was taken.

The SAFEARRAY dimension definitions are as follows:

[1st dimension] index of the process in the array

[2nd dimension] describes process properties.

1. Identifier of the process.
2. Identifier of the process that created the process being examined.
3. Memory usage, in bytes.
4. Number of execution threads started by the process
5. Base priority of any threads created by this process
6. Pointer to a null-terminated string that specifies the name of the executable file for the process. The file name does not include the path.

Return value:

This function does not return a value.

2.3. EnumApplications

Summary

Retrieves list of applications that are running on the current machine

Syntax

void EnumApplications (VARIANT* pvarApplications);

Description

Retrieves list of applications that are running on the current machine

Parameters

pvarApplications

[out]Pointer to a variant, containing two-dimensional SAFEARRAY. Each element of this SAFEARRAY is a VARIANT.

pvarApplications receives a list of all running applications and their properties. The list describes the applications residing in the system address space when a snapshot was taken.

The SAFEARRAY dimension definitions are as follows:

[1st dimension] index of the application in the array

[2nd dimension] describes application properties.

1. Identifier of the process.
2. Identifier of the thread. This identifier is compatible with the thread identifier returned by the **CreateProcess** function.
3. Process handle.
4. Handle to the icon.
5. Application status.
6. Application name.

Return value:

This function does not return a value.

2.4. EnumProcessModules

Summary

Retrieves list of modules that are used by the application. This list defines the set of files needed for the module to load and execute as a running process.

Syntax

```
void EnumProcessModules (        long lProcId,
                                 VARIANT* pvarProcModules);
```

Description

Retrieves list of modules that are used by the application. This list defines the set of files needed for the module to load and execute as a running process.

Parameters

lProcId
[in] Identifier of the process returned by EnumProcesses method.
pvarProcModules

[out]Pointer to a variant, containing two-dimensional SAFEARRAY. Each element of this SAFEARRAY is a VARIANT.

pvarProcModules receives a list of all modules in the specified process when a snapshot was taken.

The SAFEARRAY dimension definitions are as follows:

[1st dimension] index of the module in the array

[2nd dimension] describes module properties.

1. Module identifier in the context of the owning process. It is not a handle.
2. Identifier of the process to be examined.
3. Global usage count on the module.
4. Module usage count in the context of the owning process.
5. Base address of the module in the context of the owning process.
6. Size of the module, in bytes.
7. Handle to the module in the context of the owning process.
8. Pointer to a null-terminated string that specifies the module name.
9. Pointer to a null-terminated string that specifies the module path.

Return value:
This function does not return a value.

2.5. EnumProcessThreads

Summary

Retrieves list of threads executed by the process.

Syntax

```
void EnumProcessThreads (        long lProcId,
                                 VARIANT* pvarProcThreads);
```

Description

Retrieves list of threads executed by the process.

Parameters

lProcId
[in] Identifier of the process returned by EnumProcesses method.
pvarProcThreads
[out]Pointer to a variant, containing two-dimensional SAFEARRAY. Each element of this SAFEARRAY is a VARIANT.

pvarProcThreads receives a list of all threads associated with process.

The SAFEARRAY dimension definitions are as follows:

[1st dimension] index of the thread in the array

[2nd dimension] describes thread properties.

1. Thread identifier. This identifier is compatible with the thread identifier returned by the **CreateProcess** function.
2. Identifier of the process that created the thread.
3. Initial priority level assigned to a thread.
4. Change in the priority level of a thread. This value is a signed delta from the base priority level assigned to the thread
5. Number of references to the thread. A thread exists as long as its usage count is nonzero. As soon as its usage count becomes zero, a thread terminates.

Return value:

This function does not return a value.

2.6. QuerySetPriorityClass

Summary

Sets the priority class for the specified process.

Syntax

long QuerySetPriorityClass(long lProcId, long lPriority, short bSet);

Description

By default, the priority class of a process is NORMAL_PRIORITY_CLASS. Processes that monitor the system, such as screen savers or applications that periodically update a display, should use **IDLE_PRIORITY_CLASS**. This prevents the threads of this process, which do not have high priority, from interfering with higher priority threads. Use HIGH_PRIORITY_CLASS with care. If a thread runs at the highest priority level for extended periods, other threads in the system will not get processor time. If several threads are set at high priority at the same time, the threads lose their effectiveness. The high-priority class should be reserved for threads that must respond to time-critical events. If your application performs one task that requires the high-priority class while the rest of its tasks are normal priority, use QuerySetPriorityClass to raise the priority class of the application temporarily; then reduce it after the time-critical task has been completed. You should almost never use REALTIME_PRIORITY_CLASS, because this interrupts system threads that manage mouse input, keyboard input, and background disk flushing. This class can be appropriate for applications that "talk" directly to hardware or that perform brief tasks that should have limited interruptions

Parameters

lProcId
[in] Handle to the process

lPriority

[in] Specifies the priority class for the process. This parameter can be one of the predefined values:

- IDLE_PRIORITY_CLASS
- NORMAL_PRIORITY_CLASS
- HIGH_PRIORITY_CLASS
- REALTIME_PRIORITY_CLASS

bSet
[in] 1- sets the priority,
0- tests if priority can be changed

Return value:

If the function succeeds, the return value is zero.

2.7. QueryTerminateProcess

Summary

Terminates the specified process and all of its threads.

Syntax

long QueryTerminateProcess(

long lProcId,
short bTerminate);
Terminates the specified process and all of its threads.

Parameters

lProcId
[in] Handle to the process

bTerminate
[in] 1- termintes the specified process,
0- checks if process can be terminated.

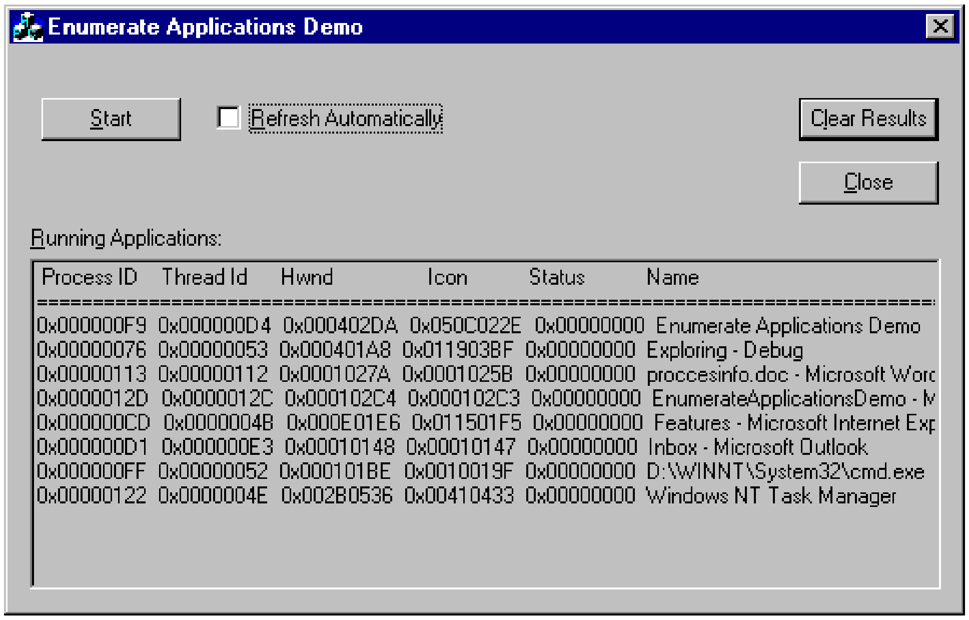
Return value:

If the function succeeds, the return value is zero

3. Examples

skprocessInfo ActiveX Control comes with complete documentation, VC++/VB/ASP sample code, and working demo programs distributed during install time.
Demo programs below were written to use SkprocessInfo control to retrieve process related information.

Figure 1 Enumerate Applications Demo



```
Process ID   Thread Id    Hwnd          Icon         Status       Name
==================================================================================
0x000000F9  0x000000D4  0x000402DA  0x050C022E  0x00000000  Enumerate Applications Demo
0x00000076  0x00000053  0x000401A8  0x011903BF  0x00000000  Exploring - Debug
0x00000113  0x00000112  0x0001027A  0x0001025B  0x00000000  proccesinfo.doc - Microsoft Worc
0x0000012D  0x0000012C  0x000102C4  0x000102C3  0x00000000  EnumerateApplicationsDemo - M
0x000000CD  0x0000004B  0x000E01E6  0x011501F5  0x00000000  Features - Microsoft Internet Exp
0x000000D1  0x000000E3  0x00010148  0x00010147  0x00000000  Inbox - Microsoft Outlook
0x000000FF  0x00000052  0x000101BE  0x0010019F  0x00000000  D:\WINNT\System32\cmd.exe
0x00000122  0x0000004E  0x002B0536  0x00410433  0x00000000  Windows NT Task Manager
```

Figure 2 Enumerate Processes Demo

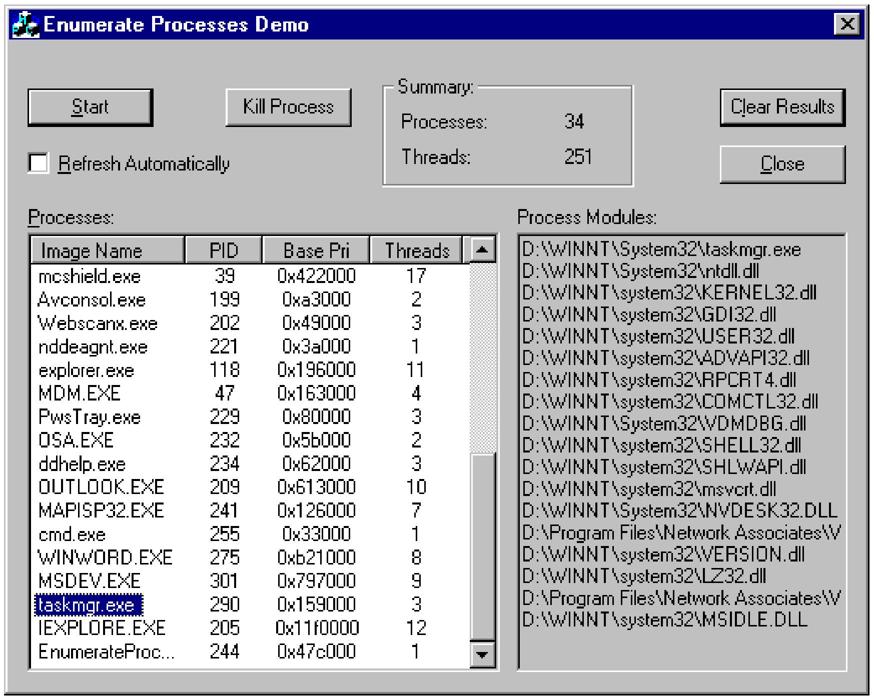
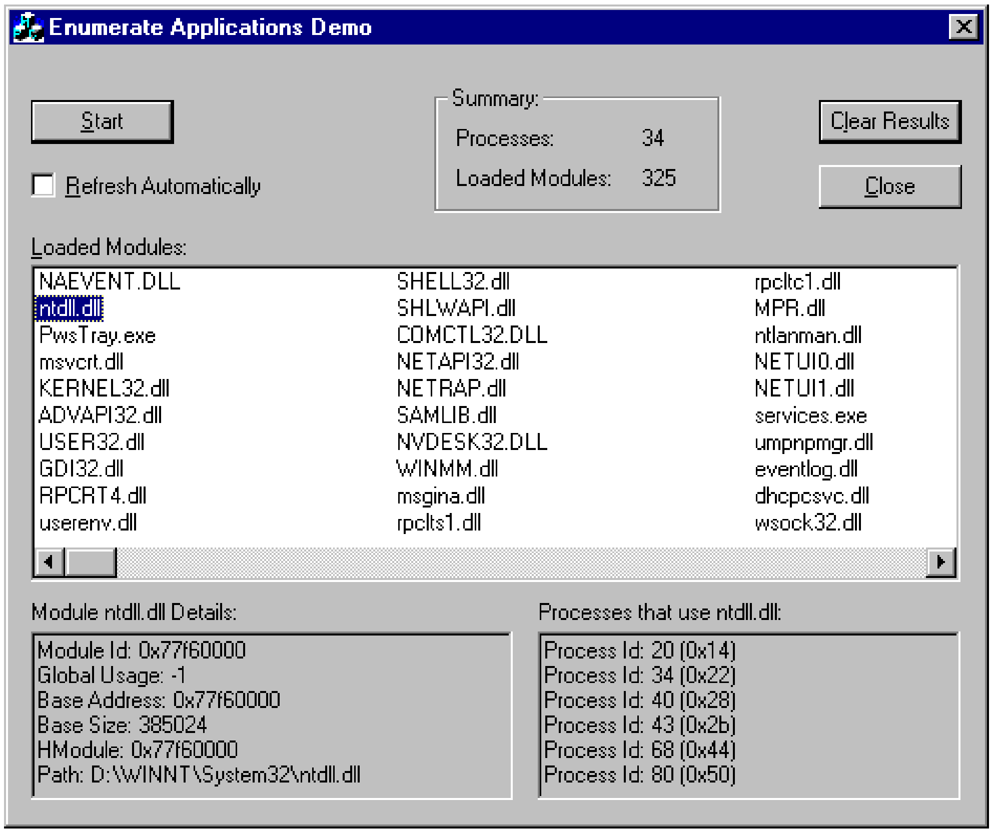


Image Name	PID	Base Pri	Threads
mcshield.exe	39	0x422000	17
Avconsol.exe	199	0xa3000	2
Webscanx.exe	202	0x49000	3
nddeagnt.exe	221	0x3a000	1
explorer.exe	118	0x196000	11
MDM.EXE	47	0x163000	4
PwsTray.exe	229	0x80000	3
OSA.EXE	232	0x5b000	2
ddhelp.exe	234	0x62000	3
OUTLOOK.EXE	209	0x613000	10
MAPISP32.EXE	241	0x126000	7
cmd.exe	255	0x33000	1
WINWORD.EXE	275	0xb21000	8
MSDEV.EXE	301	0x797000	9
taskmgr.exe	290	0x159000	3
IEXPLORE.EXE	205	0x11f0000	12
EnumerateProc...	244	0x47c000	1

```
D:\WINNT\System32\taskmgr.exe
D:\WINNT\System32\ntdll.dll
D:\WINNT\system32\KERNEL32.dll
D:\WINNT\system32\GDI32.dll
D:\WINNT\system32\USER32.dll
D:\WINNT\system32\ADVAPI32.dll
D:\WINNT\system32\RPCRT4.dll
D:\WINNT\system32\COMCTL32.dll
D:\WINNT\System32\VDMDBG.dll
D:\WINNT\system32\SHELL32.dll
D:\WINNT\system32\SHLWAPI.dll
D:\WINNT\system32\msvcrt.dll
D:\WINNT\System32\NVDESK32.DLL
D:\Program Files\Network Associates\V
D:\WINNT\system32\VERSION.dll
D:\WINNT\system32\LZ32.dll
D:\Program Files\Network Associates\V
D:\WINNT\system32\MSIDLE.DLL
```

Figure 3 Enumerate Modules Demo



Summary:
Processes: 34
Loaded Modules: 325

Loaded Modules:

```
NAEVENT.DLL        SHELL32.dll        rpcltc1.dll
ntdll.dll          SHLWAPI.dll        MPR.dll
PwsTray.exe        COMCTL32.DLL       ntlanman.dll
msvcrt.dll         NETAPI32.dll       NETUI0.dll
KERNEL32.dll       NETRAP.dll         NETUI1.dll
ADVAPI32.dll       SAMLIB.dll         services.exe
USER32.dll         NVDESK32.DLL       umpnpmgr.dll
GDI32.dll          WINMM.dll          eventlog.dll
RPCRT4.dll         msgina.dll         dhcpcsvc.dll
userenv.dll        rpclts1.dll        wsock32.dll
```

Module ntdll.dll Details:

```
Module Id: 0x77f60000
Global Usage: -1
Base Address: 0x77f60000
Base Size: 385024
HModule: 0x77f60000
Path: D:\WINNT\System32\ntdll.dll
```

Processes that use ntdll.dll:

```
Process Id: 20 (0x14)
Process Id: 34 (0x22)
Process Id: 40 (0x28)
Process Id: 43 (0x2b)
Process Id: 68 (0x44)
Process Id: 80 (0x50)
```