

# Spy++ Help

## How To Use Spy++

[Using Spy++](#)

[Starting Spy++](#)

[Working in Spy++](#)

[Using The Spy++ Toolbar](#)

[Using The Window Finder Tool](#)

[Expanding Spy++ Trees](#)

## How To Use Spy++ Views

[Using Spy++ Views](#)

[The Windows View](#)

[The Processes View](#)

[The Threads View](#)

[The Messages View](#)

[Controlling the Messages View](#)

## Copyright Notice for Spy++ Help

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without express written permission of Microsoft Corporation.

© 1993-97 Microsoft Corporation. All rights reserved.

Microsoft is a registered trademark and Windows and Windows NT are trademarks of Microsoft Corporation.

## Using Spy++

Spy++ (SPYXX.EXE) is a Win32-based utility that gives you a graphical view of the system's processes, threads, windows, and window messages. With Spy++, you can:

- Display a graphical tree of relationships among system objects, including processes, threads, and windows.
- Search for specified windows, threads, processes, or messages.
- View the properties of selected windows, threads, processes, or messages.
- Select a window, thread, process, or message directly from the view.
- Use the Finder Tool to select a window by mouse positioning.
- Set message options using complex message log selection parameters.

**Tip** While using Spy++, in many instances you can click the right mouse button to display a pop-up menu of frequently used commands. The commands available depend on where the pointer is. For example, if you click while pointing at an entry in the window view and the selected window is visible, the pop-up **Highlight** menu item will cause the border of the selected window to flash so that the window can be easily located on the screen.

## Working In Spy++

Spy++ has a toolbar and hyperlinks to help you work faster. It also provides a **Refresh** command to update the active view, a Window Finder Tool to make spying easier, and a **Font** dialog box to customize view windows. Additionally, Spy++ saves and restores user preferences.

There are two utilities similar to Spy++: PView, which show details on processes and threads, and DDESPY.EXE, a monitoring program for Dynamic Data Exchange (DDE) messages.

## Starting Spy++

### ► To start Spy++

- From the **Tools** menu, choose **Spy++**.

**Note** You can run only one copy of Spy++. Attempting to run additional copies of Spy++ will bring the currently running Spy++ to the front.













Spy++ is a read-only program. Using Spy++ does not change program operation, but can slow program execution.

## Viewing with Spy++

When Spy++ starts, it opens a window titled “Windows 1,” which shows a tree view of all windows and controls in the system. There are also three other views available in Spy++: Messages, Processes, and Threads.

## The Spy++ Toolbar

The toolbar appears beneath the menu bar. It provides shortcut commands for opening new views, starting or stopping the message stream display, changing message stream options, clearing a message stream window, and finding a window. You can display or hide the toolbar with the **Toolbar** command on the **View** menu.

Button	Effect
	Creates a window to display a tree view of all windows and controls in the system. See <a href="#">The Windows View</a> .
	Creates a window to display a tree view of all processes in the system. See <a href="#">The Processes View</a> .
	Creates a window to display a tree view of all threads in the system. See <a href="#">The Threads View</a> .
	Creates a window to display window messages. This button calls up the <b>Message Options</b> dialog box to let you select the window whose messages will be displayed in addition to other options. See <a href="#">The Messages View</a> .
	Starts the message logging and displays the message stream. This button is available only when a Messages window is active (has the focus). See <a href="#">Starting and Stopping the Message Log Display</a> .
	Stops the message logging and display of the message stream. This button is available only when a Messages window is active (has the focus). See <a href="#">Starting and Stopping the Message Log Display</a> .
	Displays the <b>Message Options</b> dialog box. Use this dialog box to select windows and message types for viewing. This button is available only when a Messages window is active (has the focus). See <a href="#">Choosing Message Options</a> .
	Clears the contents of the active Messages window. This button is available only when a Messages window is active (has the focus).
	Opens the <b>Find Window</b> dialog box, which lets you select a window to view messages or find properties. See <a href="#">The Window Finder Tool</a> .
	Searches the current view for a matching window, process, thread, or message.
	Searches the current view for the next matching window, process, thread, or message. This button (and the related menu item) is available only when there is a valid search result that is not unique. For example, when you use a window handle as the search criteria in the window tree, it will produce unique results, since there is only one window with that handle in the window tree. In this instance, Find Next is not available.
	Searches the current view for the previous matching window, process, thread, or message. This button (and the related menu item) is available only when there is a valid search result that is not unique. For example, when you use a window handle as the search criteria in the window tree, it will produce unique results, since there is only one window with that handle in the window

tree. In this instance, Find Previous is not available.



## Refreshing the View

Spy++ takes a “snapshot” of the system tables and refreshes a view based on this information. It is important that you periodically refresh your system views. If you have a Spy++ view open and have not refreshed the view, you will not see those processes, threads, and windows that are subsequently created. Also, you may see items that no longer exist. The **Refresh** command is available for all views except the Messages view.

- ▶ **To refresh the currently active view**
  - Choose **Refresh** from the **Window** menu.

## Changing Fonts

You can change the font, font style, and font size for Spy++ windows.

► **To change font options**

- 1 From the **View** menu, choose **Font**.
- 2 Choose a font, font style, and font size.
- 3 Click **OK**.

Selecting **Save Font As Default** will cause all future Spy++ windows to use this font.

## Expanding and Collapsing Spy++ Trees

You can expand and collapse the Windows, Processes, and Threads views using two methods: by clicking on the icons in the window or by using the **Tree** menu. The + and – icons in the tree act as they do in the Developer Studio project window.

The **Tree** menu contains four commands:

Menu Command	Description
<b>Expand One Level</b>	Expands the currently selected item to the next level.
<b>Expand Branch</b>	Fully expands the currently selected item.
<b>Expand All</b>	Fully expands all items in the window.
<b>Collapse</b>	Fully collapses the currently selected item.

**Tip** If you expand a process, you see all the threads the process owns. If you expand a thread, you see a list of all the windows it owns.

### ► To expand or collapse Spy++ trees

- 1 Highlight one of the items in the Windows, Processes, or Threads view.
- 2 From the **Tree** menu, choose one of the expand or collapse commands.

## The Windows View

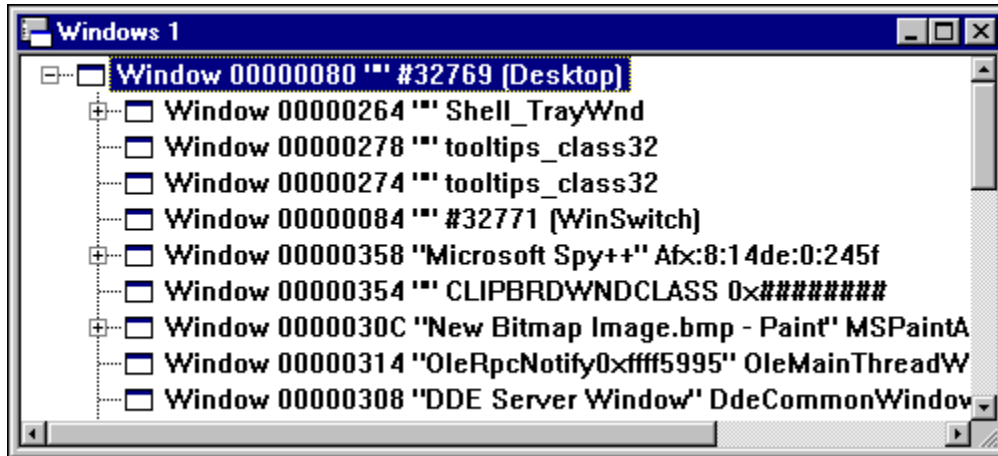
The Windows view displays a tree of all windows and controls in the system. The Windows view shows only windows.

► **To open the Windows view**

- From the **Spy** menu, choose **Windows**.

The figure below shows the Spy++ representation of the Windows view with the first level expanded.

### The Spy++ Windows View



The current desktop window is at the top of the tree. All other windows are children of the desktop, and are listed according to the standard window hierarchy, with sibling windows ordered by Z-order. You can collapse or expand the whole tree by clicking the – or + symbol for the top level window.

The Windows view is most useful if you need to find a particular window. If you start with a tree expanded at the second level (all windows that are children of the desktop), then you can identify a desired desktop-level window by its class name and title. Once you have found the desktop-level window, you can expand the level to find a specific child window.

## The Window Finder Tool

With the Window Finder Tool you can find the properties of a selected window. The Finder Tool can find disabled child windows and discern which window to highlight if disabled child windows overlap.

### ► To find a window to spy on

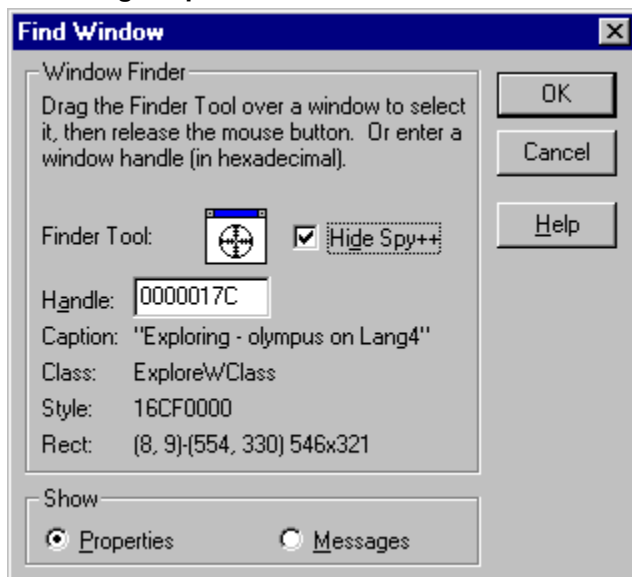
- 1 Arrange your windows so that Spy++ and the subject window are visible.
- 2 From the **Spy** menu, choose **Find Window** to open the **Find Window** dialog box.
- 3 Drag the **Finder Tool** to the desired window. As you drag the tool, window details display in the dialog box.

– or –

If you know the handle of the window you want (for example, from the debugger), type it in the **Handle** box.

- 4 Under **Show**, choose **Properties** or **Messages** to select what kind of information to display.
- 5 Click **OK**.

### Showing Properties with the Find Window



**Tip** To reduce screen clutter, select the **Hide Spy** option on the **Find Window** dialog box. This option removes the main Spy++ window and leaves the **Find Window** dialog box visible on top of your other applications. The Spy++ main window will be restored when you click **OK** or **Cancel**, or when you clear the **Hide Spy++** option.

## Searching for a Window

You can search for a specific window by using its handle, caption, class, or a combination of its caption and class as search criteria. You can also specify the initial direction of the search. The fields in the dialog box will show the attributes of the selected window in the window tree.

► **To search for a window**

- 1 Arrange your windows so that Spy++, an active Windows view, and the subject window are visible.
- 2 From the **Search** menu, choose **Find Window**.
- 3 Drag the **Finder Tool** to the desired subject window. As you drag the tool, window details display in the dialog box.

– or –

If you know the handle of the window you want (for example, from the debugger), type it in the **Handle** box.

– or –

If you know the caption and/or class of the window you want, type them in the **Caption** and class box and delete the **Handle** box.

- 4 Choose **Up** or **Down** for the initial direction of the search.
- 5 Click **OK**.

If a matching window is found, it is highlighted in the Windows view window. See [The Windows View](#).

**Tip** To reduce screen clutter, select the **Hide Spy** option on the **Find Window** dialog box. This option removes the main Spy++ window and leaves the **Find Window** dialog box visible on top of your other applications. The Spy++ main window will be restored when you click **OK** or **Cancel**, or when you clear the **Hide Spy++** option.

## Opening Window Properties

You can find out more about entries in the Windows view with the **Window Properties** dialog box.

► **To open a Window Properties dialog box**

- Click the item, then choose **Properties** from the **View** menu.

Properties dialog boxes are not modal, so you can click on another item in a view window and the dialog box will show information on the selected item.

## Window Properties

The **Window Properties** dialog box contains five tabs:

- General
- Styles
- Windows
- Class
- Process



## Window Properties General Tab

The options on the **Window Properties General** tab are:

Entry	Description
<b>Window Caption</b>	The text in the window caption, or text contained in a window if it is a control.
<b>Window Handle</b>	The unique ID of this window. Window handle numbers are reused; they identify a window only for the lifetime of that window.
<b>Window Proc</b>	The virtual address of the window procedure function for this window. This field also indicates whether this window is a Unicode window, and whether it is subclassed.
<b>Rectangle</b>	The bounding rectangle for the window. The size of the rectangle is also displayed. Units are pixels in screen coordinates.
<b>Restored Rect</b>	The bounding rectangle for the restored window. The size of the rectangle is also displayed. Restored Rect will differ from Rectangle only when the window is maximized or minimized. Units are pixels in screen coordinates.
<b>Client Rect</b>	The bounding rectangle for the window client area. The size of the rectangle is also displayed. Units are pixels relative to the top left of the window client area.
<b>Instance Handle</b>	The instance handle of the application. Instance handles are not unique.
<b>Control ID or Menu Handle</b>	If the window being displayed is a child window, the Control ID label is displayed. Control ID is an integer that identifies this child window's control ID. If the window being displayed is not a child window, the Menu Handle label is displayed. Menu Handle is an integer that identifies the handle of the menu associated with this window.
<b>User Data</b>	Application-specific data that is attached to this window structure.
<b>Window Bytes</b>	The number of extra bytes associated with this window. The meaning of these bytes is determined by the application. Expand the list box to see the byte values in DWORD format.

## Window Properties Styles Tab

The options on the **Window Properties Styles** tab are:

<b>Entry</b>	<b>Description</b>
<b>Window Styles</b>	A combination of window style codes and control styles if applicable.
<b>Extended Styles</b>	A combination of extended window style codes.

## Window Properties Windows Tab

The options on the **Window Properties Windows** tab are:

Entry	Description
<b>Next Window</b>	The handle of the next sibling window in the same sequence (Z-order) shown in the window tree view (“none” if there is no next window). Click on this entry to view the properties of the next window.
<b>Previous Window</b>	The handle of the previous sibling window in the same sequence (Z-order) shown in the window tree view (“none” if there is no previous window). Click on this entry to view the properties of the previous window.
<b>Parent Window</b>	The handle of this window’s parent window (“none” if there is no parent). Click on this entry to view the properties of the parent window.
<b>First Child</b>	The handle of this window’s first child window, in the sequence (Z-order) shown in the window tree view (“none” if there are no child windows). Click on this value to view the properties of the first child window.
<b>Owner Window</b>	The handle of this window’s owner window. An application’s main window typically owns system-modal dialog windows, for example (“none” if there is no owner). Click on this entry to view the properties of the owner window.

## Window Properties Class Tab

The options on the **Window Properties Class** tab are:

Entry	Description
<b>Class Name</b>	The name (or ordinal number) of this window class.
<b>Class Styles</b>	A combination of class style codes.
<b>Class Bytes</b>	Application-specific data associated with this window class.
<b>Class Atom</b>	The atom for the class returned by the <b>RegisterClass</b> call.
<b>Instance Handle</b>	The instance handle of the module that registered the class. Instance handles are not unique.
<b>Window Bytes</b>	The number of extra bytes associated with each window of this class. The meaning of these bytes is determined by the application. Expand the list box to see the byte values in DWORD format.
<b>Window Proc</b>	The current address of the <b>WndProc</b> function for windows of this class. This differs from <b>Window Proc</b> on the <b>General</b> tab if the window is subclassed.
<b>Menu Name</b>	The name of the main menu that is associated with windows of this class ("none" if there is no menu).
<b>Icon Handle</b>	The handle for the icon that is associated with windows of this class ("none" if there is no icon).
<b>Cursor Handle</b>	The handle for the cursor that is associated with windows of this class ("none" if there is no cursor).
<b>Bkgnd Brush</b>	The handle for the background brush that is associated with windows of this class, or one of the predefined COLOR_* colors for painting the window background ("none" if there is no brush).

## Window Properties Process Tab

The options on the **Window Properties Process** tab are:

Entry	Description
<b>Process ID</b>	The ID of the process that owns the thread that created this window. Click on this value to view the properties of this process.
<b>Thread ID</b>	The ID of the thread that created this window. Click on this value to view the properties of this thread.

## The Processes View

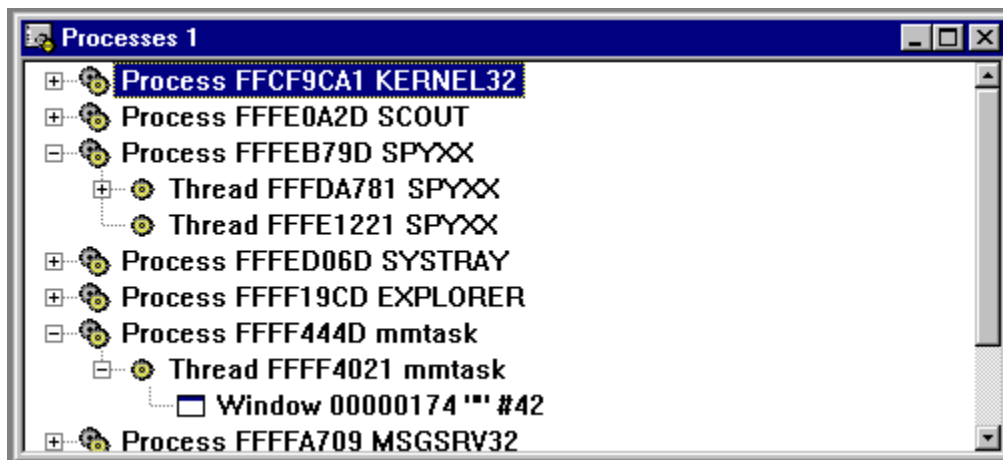
Microsoft Windows supports multiple processes. Each process can have one or more threads, and each thread can have one or more associated top-level windows. Each top-level window can own a series of windows. A + symbol indicates that a level is already collapsed. Click the + symbol to expand the level. The collapsed view consists of one line per process.

Use the Processes view if you want to examine a particular system process, which usually corresponds to an executing program. Processes are identified by module names, or they are designated “system processes.” To find a process, collapse the tree and search the list.

- ▶ **To open the Processes view**
  - From the **Spy** menu, choose **Processes**.

The Processes View window appears as shown in the following figure.

**The Processes View Window**



## Searching for a Process

You can search for a specific process by using its process ID or module string as search criteria. You can also specify the initial direction of the search. The fields in the dialog box will show the attributes of the selected process in the process tree.

### ► To search for a process

- 1 Arrange your windows so that Spy++ and an active Processes view are visible.
- 2 From the **Search** menu, choose **Find Process** to open the **Process Search** dialog box.
- 3 Type the process ID or a module string as search criteria.
- 4 Choose **Up** or **Down** for the initial direction of the search.
- 5 Click **OK**.

If a matching process is found, it is highlighted in the The Processes View window.

**Tip** To find all the processes owned by a module, erase the **Process** box and type the module name in the **Module** box. Then use **Find Next** to continue searching for processes.

## Opening Process Properties

You can find out more about entries in the Processes view with the **Process Properties** dialog box.

- ▶ **To open a Process Properties dialog box**
  - Click the item, then choose **Properties** from the **View** menu.

Properties dialog boxes are not modal, so you can click on another item in a view window and the dialog box will show information on the selected item.



## Process Properties for Windows 95

For Windows 95, the **Process Properties** dialog box contains only the General tab.

## Process Properties General Tab (Windows 95)

The options on the **Process Properties General** tab for Windows 95 are:

Entry	Description
<b>Module Name</b>	The name of the module.
<b>Process ID</b>	The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data.
<b>Priority Base</b>	The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process's base priority.
<b>Threads</b>	The number of threads currently active in this process.

## Process Properties for Windows NT

For Windows NT, the **Process Properties** dialog box contains four tabs:

- General
- Memory
- Page File
- Space

## Process Properties General Tab (Windows NT)

The options on the **Process Properties General** tab for Windows NT are:

Entry	Description
<b>Module Name</b>	The name of the module.
<b>Process ID</b>	The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data.
<b>Priority Base</b>	The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process's base priority.
<b>Threads</b>	The number of threads currently active in this process.
<b>CPU Time</b>	Total CPU time spent on this process and its threads. Equal to User Time plus Privileged Time.
<b>User Time</b>	The cumulative elapsed time that this process's threads have spent executing code in User Mode in non-idle threads. Applications execute in User Mode, as do subsystems such as the window manager and the graphics engine.
<b>Privileged Time</b>	The total elapsed time this process has been running in Privileged Mode in non-idle threads. The service layer, the Executive routines, and the Kernel execute in Privileged Mode. Device drivers for most devices other than graphics adapters and printers also execute in Privileged Mode. Some work that Windows does for your application may appear in other subsystem processes in addition to Privileged Time.
<b>Elapsed Time</b>	The total elapsed time this process has been running.

## Process Properties Memory Tab (Windows NT)

The options on the **Process Properties Memory** tab for Windows NT are:

Entry	Description
<b>Virtual Bytes</b>	The current size (in bytes) of the virtual address space the process is using. The use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. However, virtual space is finite, and using too much may limit the ability of the process to load libraries.
<b>Peak Virtual Bytes</b>	The maximum number of bytes of virtual address space the process has used at any one time.
<b>Working Set</b>	The set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from the Working Set. If they are needed, they will be soft-faulted back into the Working Set before they leave main memory.
<b>Peak Working Set</b>	The maximum number of pages in the working set of this process at any point in time.
<b>Paged Pool Bytes</b>	The current amount of paged pool the process has allocated. Paged pool is a system memory area where operating system components acquire space as they accomplish their appointed tasks. Paged pool pages can be paged out to the paging file when not accessed by the system for sustained periods of time.
<b>Nonpaged Pool Bytes</b>	The current number of bytes in the nonpaged pool allocated by the process. The nonpaged pool is a system memory area where space is acquired by operating system components as they accomplish their appointed tasks. Nonpaged pool pages cannot be paged out to the paging file; they remain in main memory as long as they are allocated.
<b>Private Bytes</b>	The current number of bytes this process has allocated that cannot be shared with other processes.
<b>Free Bytes</b>	The total unused virtual address space of this process.
<b>Reserved Bytes</b>	The total amount of virtual memory reserved for future use by this process.
<b>Free Image Bytes</b>	The amount of virtual address space that is not in use or reserved by images within this process.
<b>Reserved Image Bytes</b>	The sum of all virtual memory reserved by images run within this process.

## Process Properties Page File Tab (Windows NT)

The options on the **Process Properties Page File** tab for Windows NT are:

Entry	Description
<b>Page File Bytes</b>	The current number of pages that this process is using in the paging file. The paging file stores pages of data used by the process but not contained in other files. The paging file is used by all processes, and lack of space in the paging file can cause errors while other processes are running.
<b>Peak Page File Bytes</b>	The maximum number of pages that this process has used in the paging file.
<b>Page Faults</b>	The number of Page Faults by the threads executing in this process. A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. Thus, the page will not be retrieved from disk if it is on the standby list and hence already in main memory, or if it is being used by another process with which the page is shared.

## Process Properties Space Tab (Windows NT)

The options on the **Process Properties Space** tab for Windows NT are:

Entry	Description
<b>Show for space marked as</b>	Use this list box to select the category of space (image, mapped, reserved, or unassigned).
<b>Executable Bytes</b>	For the selected category, the sum of all the address space that this process is using. Executable memory is memory that can be executed by programs, but may not be read or written.
<b>Exec-Read-Only Bytes</b>	For the selected category, the sum of all the address space in use with read-only properties that this process is using. Exec-read-only memory is memory that can be executed as well as read.
<b>Exec-Read-Write Bytes</b>	For the selected category, the sum of all the address space in use with read-write properties that this process is using. Exec-read-write memory is memory that can be executed by programs as well as read and modified.
<b>Exec-Write Copy Bytes</b>	For the selected category, the sum of all the address space that can be executed by programs as well as read and written. This type of protection is used when memory needs to be shared between processes. If the sharing processes only read the memory, then they will all use the same memory. If a sharing process desires write access, then a copy of this memory will be made for the process.
<b>No-Access Bytes</b>	For the selected category, the sum of all the address space that prevents a process from using it. An access violation is generated if writing or reading is attempted.
<b>Read-Only Bytes</b>	For the selected category, the sum of all the address space that can be executed as well as read.
<b>Read-Write Bytes</b>	For the selected category, the sum of all the address space that allows reading and writing.
<b>Write-Copy Bytes</b>	For the selected category, the sum of all the address space that allows memory sharing for reading but not for writing. When processes are reading this memory, they can share the same memory. However, when a sharing process wants to have read/write access to this shared memory, a copy of that memory is made for writing.

## The Threads View

The Threads view is a flat listing of all threads with associated windows. Processes are not included. You can easily find the process that owns a selected thread. Use the Threads view to search for a particular thread.

► **To open the Threads view**

- From the **Spy** menu, choose **Threads**.

## Searching for a Thread

You can search for a specific thread by using its thread ID or module string as search criteria. You can also specify the initial direction of the search. The fields in the dialog box will show the attributes of the selected thread in the thread tree.

### ► To search for a thread

- 1 Arrange your windows so that Spy++ and an active Threads view are visible.
- 2 From the **Search** menu, choose **Find Thread**.
- 3 Type the thread ID or a module string as search criteria.
- 4 Choose **Up** or **Down** for the initial direction of the search.
- 5 Click **OK**.

If a matching thread is found, it is highlighted in the Threads view window.

**Tip** To find all the threads owned by a module, erase the Thread box and type the module name in the **Module** box. Then use **Find Next** to continue searching for processes.



## Opening Thread Properties

You can find out more about entries in the [Threads view](#) with the **Thread Properties** dialog box.

► **To open a Thread Properties dialog box**

- Click the item, then choose **Properties** from the **View** menu.

Properties dialog boxes are not modal, so you can click on another item in a view window and the dialog box will show information on the selected item.

## Thread Properties for Windows 95

For Windows 95, the **Thread Properties** dialog box contains only the General tab.

## Thread Properties General Tab (Windows 95)

The options on the **Thread Properties General** tab for Windows 95 are:

Entry	Description
<b>Module Name</b>	The name of the module.
<b>Thread ID</b>	The unique ID of this thread. Note that Thread ID numbers are reused; they identify a thread only for the lifetime of that thread.
<b>Process ID</b>	The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data. Click on this value to view the properties of the process ID.
<b>Current Priority</b>	The current dynamic priority of this thread. Threads within a process can raise and lower their own base priority relative to the base priority of the process.
<b>Base Priority</b>	The current base priority of this thread.

## Thread Properties for Windows NT

For Windows NT, the **Thread Properties** dialog box contains only the General tab.

## Thread Properties General Tab (Windows NT)

The options on the **Thread Properties General** tab for Windows NT are:

Entry	Description
<b>Module Name</b>	The name of the module.
<b>Thread ID</b>	The unique ID of this thread. Note that thread ID numbers are reused; they identify a thread only for the lifetime of that thread.
<b>Process ID</b>	The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data. Click on this value to view the properties of the process ID.
<b>Thread State</b>	The current state of the thread. A Running thread is using a processor; a Standby thread is about to use one. A Ready Thread is waiting to use a processor because one is not free. A thread in Transition is waiting for a resource to execute, such as waiting for its execution stack to be paged in from disk. A Waiting thread does not need the processor because it is waiting for a peripheral operation to complete or a resource to become free.
<b>Wait Reason</b>	This is applicable only when the thread is in the Wait state. Event Pairs are used to communicate with protected subsystems.
<b>CPU Time</b>	Total CPU time spent on this process and its threads. Equal to User Time + Privileged Time.
<b>User Time</b>	The total elapsed time that this thread has spent executing code in User Mode. Applications execute in User Mode, as do subsystems like the window manager and the graphics engine.
<b>Privileged Time</b>	The total elapsed time that this thread has spent executing code in Privileged Mode. When a Windows system service is called, the service will often run in Privileged Mode to gain access to system-private data. Such data is protected from access by threads executing in User Mode. Calls to the system may be explicit, or they may be implicit, such as when a page fault or an interrupt occurs.
<b>Elapsed Time</b>	The total elapsed time (in seconds) this thread has been running.
<b>Current Priority</b>	The current dynamic priority of this thread. Threads within a process can raise and lower their own base priority relative to the base priority of the process.
<b>Base Priority</b>	The current base priority of this thread.
<b>Start Address</b>	Starting virtual address for this thread.
<b>User PC</b>	The user program counter for the thread.
<b>Context Switches</b>	The number of switches from one thread to another. Thread switches can occur either inside a single process or across processes. A thread switch may be

caused by one thread asking another for information, or by a thread being preempted when a higher priority thread becomes ready to run.

## The Messages View

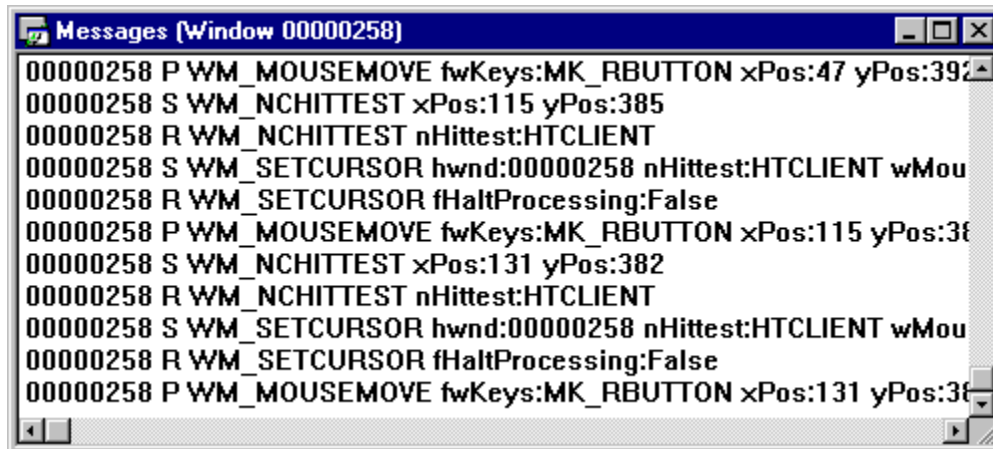
Each window has an associated message stream. You can view this message stream in the Messages view. You can create a Messages view for a thread or process as well. This allows you to view messages sent to all windows owned by a specific process or thread, which is particularly useful for capturing window initialization messages.

### ► To open the Messages view for a window, process, or thread

- 1 Move the focus to either a Windows, Processes, or Threads View window.
- 2 Find the name of the window, process, or thread that you want to examine, and select it.
- 3 From the **Spy** menu, choose **Messages**.
- 4 Select the message options you want.
- 5 Click **OK** to begin logging messages.
- 6 From the **Messages** menu, choose **Stop Logging**.

The Messages view appears as shown below. Note that the first column contains the window handle, and the second column contains a message code (explained in [Message Codes](#)). Decoded message parameters and return values are on the right.

### The Messages View Window



For a more direct way to select a window for message stream viewing, use the following procedure.

### ► To open the Messages view for a visible window using the Message Options dialog box

- 1 Arrange your windows so that both Spy++ and the subject window are visible.
- 2 From the **Spy** menu, choose **Messages**.
- 3 From the **Windows** tab, drag the **Finder Tool** to the window you want. As you drag the tool, window details appear in the dialog box.
- 4 Click **Ok** to begin logging messages.
- 5 From the **Messages** menu, choose **Stop Logging**.

**Tip** To reduce screen clutter, select the **Hide Spy** option in the **Find Window** dialog box. This removes the main Spy++ window and leaves the **Find Window** dialog box visible on top of your other applications. The Spy++ main window will be restored when you click **OK** or **Cancel**, or by clearing the **Hide Spy** option.

## Message Codes

Each message line shown in the Messages view contains a 'P,' 'S,' 's,' or 'R' code. These codes have the following meanings:

Code	Meaning
P	The message was posted to the queue with the <b>PostMessage</b> function. No information is available concerning the ultimate disposition of the message.
S	The message was sent with the <b>SendMessage</b> function. This means that the sender doesn't regain control until the receiver processes and returns the message. The receiver can, therefore, pass a return value back to the sender.
s	The message was sent, but security prevents access to the return value.
R	Each 'S' line has a corresponding 'R' (return) line that lists the message return value. Sometimes message calls are nested, which means that one message handler sends another message.



## Controlling the Messages View

With Spy++, you have considerable control over the content of the Messages view. You can start and stop displaying the messages at any time, and you can specify:

- Which message types you want to see.
- Which windows you want to monitor.
- The display format for message lines.

These settings are available from the Message Options dialog box and apply only to the selected Messages view.

## Starting and Stopping the Message Log Display

When a Messages View window is active, a **Start** or **Stop Logging** choice appears on the Spy++ **Messages** menu, and the **Start** or **Stop Logging** toolbar button becomes active.

- ▶ **To stop the message log display**
  - From the **Messages** menu, choose **Stop Logging**.
- ▶ **To start the message log display**
  - From the **Messages** menu, choose **Start Logging**.

## Choosing Message Options

The **Options** command on the **Messages** menu opens the **Message Options** dialog box with three tabs: **Windows**, **Messages**, and **Output**.

## The Windows Tab

The **Windows** tab on the **Message Options** dialog box contains the [Window Finder Tool](#). Other options on the **Windows** tab include:

Option	Description
<b>Parent</b>	Display messages for the selected window and its immediate parent window.
<b>Children</b>	Display messages for the selected window and all its child windows, including nested child windows.
<b>Windows of Same Thread</b>	Display messages for the selected window and all other windows owned by the same thread.
<b>Windows of Same Process</b>	Display messages for the selected window and all other windows owned by the same process.
<b>All Windows in System</b>	Display messages for all windows.
<b>Save Settings as Default</b>	Save the preceding settings for new message stream windows. These settings are saved when Spy++ quits.

## The Messages Tab

You can use the **Messages** tab on the **Message Options** dialog box to select message types for viewing. Typically, you first select message groups, and then fine-tune the selection by selecting individual messages. The **All** button selects all message types, and the **None** button clears all types.

Note that three entries under **Message Groups** do not map to specific entries under **Messages To View**. These include:

- WM\_USER: with a code greater than or equal to WM\_USER
- Registered: registered with the **RegisterWindowMessage** call
- Unknown: unknown messages in the range 0 to (WM\_USER – 1)

If you select these “groups,” the selection is applied directly to the message stream.

When you create a new Messages window, it can display all messages. When you filter messages from the **Messages** tab, that filter only applies to new messages, not messages that have already been displayed in the Windows view.

A grayed check box within **Message Groups** indicates that the **Messages To View** list box has been modified for messages in that group; not all of the message types in that group are selected.

If you select **Save Settings as Default**, the current settings are saved for later use as message search options. These settings are also saved when exiting Spy++.

## The Output Tab

You can use the **Output** tab on the **Message Options** dialog box to select the following options:

Option	Description
<b>Message Nesting Level</b>	Prefix nested messages with one period per level.
<b>Raw Message Parameters</b>	Display the hexadecimal <b>wParam</b> and <b>lParam</b> values.
<b>Decoded Message Parameters</b>	Display the results of message-specific decoding of the <b>wParam</b> and <b>lParam</b> values.
<b>Raw Return Values</b>	Display the hexadecimal <b>lResult</b> return value.
<b>Decoded Return Values</b>	Display the results of message-specific decoding of the <b>lResult</b> return value.
<b>Message Origin Time</b>	The elapsed time since the Windows system was started (for posted messages only).
<b>Message Mouse Position</b>	The screen coordinates of the mouse when the message was posted (for posted messages only).
<b>Lines Maximum</b>	Limit the number of lines that are retained in the currently selected Messages view.
<b>Also Log Messages to File</b>	Specify an output file for the message log. This output file is written simultaneously with the message log window.
<b>Save Settings as Default</b>	Save the preceding settings for new message stream windows. These settings are saved when you quit Spy+ +.

## Searching for a Message

You can search for a specific message by using its handle, type, or message ID as search criteria. Any one of these—or a combination—will be valid search criteria. The initial direction of the search can also be specified. The fields in the **Message Search** dialog box will be preloaded with the attributes of the selected message in the message log.

### Message Search Dialog Box



#### ► To search for a message

- 1 Arrange your windows so that Spy++ and an active Messages view are visible.
- 2 From the **Search** menu, choose **Find Message** to open the **Message Search** dialog box.
- 3 Drag the **Finder Tool** to the desired subject window. As you drag the tool, Window details display in the dialog box.
  - or –
  - If you know the window handle of the message you want, type it in the **Handle** box.
  - or –
  - If you know the message type and/or message ID you want, make a selection using the **Type** and **Message** boxes and delete the entry in the **Handle** box. You can clear any fields for which you do not want to specify values.
- 4 Type the message type or message ID to search for.
- 5 Choose **Up** or **Down** for the initial direction of the search.
- 6 Click **OK**.

If a matching message is found, it is highlighted in the Messages view window. See [The Messages View](#).

**Tip** To reduce screen clutter, select the **Hide Spy** option on the **Find Window** dialog box. This option removes the main Spy++ window and leaves the **Find Window** dialog box visible on top of your other applications. The Spy++ main window will be restored when you click **OK** or **Cancel**, or when you clear the **Hide Spy++** option.

## Opening Message Properties

You can find out more about entries in the Messages view with the **Message Properties** dialog box.

- ▶ **To open a Message Properties dialog box**
  - Click the item, then choose **Properties** from the **View** menu.



## Message Properties

The **Message Properties** dialog box contains only the General tab.

## Message Properties General Tab

The options on the **Message Properties General** tab are:

Entry	Description
<b>Window Handle</b>	The unique ID of this window. Window handle numbers are reused; they identify a window only for the lifetime of that window. Click this value to view the properties of this window.
<b>Nesting Level</b>	Depth of nesting of this message, where 0 is no nesting.
<b>Message</b>	Number, status, and name of the selected windows message.
<b>IResult</b>	The value of the <i>IResult</i> parameter, if any.
<b>wParam</b>	The value of the <i>wParam</i> parameter, if any.
<b>IParam</b>	The value of the <i>IParam</i> parameter, if any. This value is decoded if it is a pointer to a string or structure.

## **Spy Menu Commands**

The **Spy** menu provides commands to work with Spy++ Windows and views.

### **Messages**

Opens a new Messages view and opens the **Message Options** dialog box for the view.

### **Windows**

Opens a new Windows view.

### **Processes**

Opens a new Processes view.

### **Threads**

Opens a new Threads view.

### **Find Window**

Displays the **Find Window** dialog box to select a window with the mouse.

### **Exit**

Exits Spy++

## **Tree Menu Commands**

The **Tree** menu provides commands to expand or collapse views.

### **Expand One Level**

Expands the currently selected item in a system view by displaying one level beneath it.

### **Expand Branch**

Expands all levels beneath the currently selected item.

### **Expand All**

Expands and displays all levels in a system view.

### **Collapse**

Collapses all levels beneath the currently selected item.

## **Search Menu Commands**

The **Search** menu provides commands to search for specific windows.

### **Find Window/Process/Thread/Message**

Opens the **Window/Process/Thread/Message Search** dialog box, where you can use the **Finder Tool** or specify search criteria.

### **Find Next**

Searches the current view for the next matching window, process, thread, or message. This menu item (and the related toolbar button) is available only when there is a valid search result that is not unique.

### **Find Previous**

Searches the current view for the previous matching window, process, thread, or message. This menu item (and the related toolbar button) is available only when there is a valid search result that is not unique.

## **View Menu Commands**

The **View** menu provides commands for controlling the Spy++ user interface.

### **Font**

Opens the **Font** dialog box, where you can select a font for use in the current Spy++ view.

### **Toolbar**

Hides or displays the Toolbar.

### **Status Bar**

Hides or displays the Status Bar.

### **Properties**

Opens the **Properties** dialog box for the active view.

## **Message Menu Commands**

The **Messages** menu provides commands for controlling message views.

### **Start/Stop Logging**

Starts or stops the message view.

### **Options**

Opens the **Message Options** dialog box, where you can specify message search criteria.

### **Clear**

Clears the contents of the active Messages view.

## **Window Menu Commands**

The **Window** menu provides commands to manage Spy++ windows.

### **Cascade**

Rearranges all windows in a cascade pattern.

### **Tile Horizontally**

Rearrange all windows in a horizontal tile pattern.

### **Tile Vertically**

Rearranges all windows in a vertical tile pattern.

### **Arrange Icons**

Rearranges the view icons, if windows have been minimized.

### **Refresh**

Refreshes the currently selected view.

### **Window**

Allows selection of a specific window.



## **Help Menu Commands**

The **Help** menu provides commands to open help files.

### **Index**

Opens the contents screen for Spy++ Help.

### **Using Help**

Opens a Help file that describes how to use help.

### **About Spy++**

Shows the software version and copyright information for Spy++.

## **About Microsoft Spy++**

Use this dialog box to display copyright, version, ownership, and other information about Spy++.

## **Font**

Use this dialog box to select the font and font size for Spy++ views.

### **Font**

The character font for all Spy++ views.

### **Font Style**

The font style for all Spy++ views.

### **Size**

The font size for all Spy++ views.

### **Save Font as Default**

Saves the font selection for future Spy++ views. These settings are also saved when exiting Spy++.

## **Find Window**

Use this dialog box to specify window search criteria.

### **Finder Tool**

The drag and drop tool used to select a window.

### **Hide Spy++**

Removes the main Spy++ window and leaves the **Window Search** dialog box visible on top of your other applications.

### **Handle**

The unique ID of the window to search for.

### **Caption**

The text in the window caption.

### **Class**

The class of the window.

### **Style**

The style of the window.

### **Rect**

The bounding rectangle for the window.

## **Process Search**

Use this dialog box to specify process search criteria.

### **Process**

The process ID to search for.

### **Module**

The module string to search for.

### **Search Direction Up or Down**

The initial direction of the search.

## **Thread Search**

Use this dialog box to specify thread search criteria.

### **Thread**

The thread ID to search for.

### **Module**

The module string to search for.

### **Search Direction Up or Down**

The initial direction of the search.

## **Message Search**

Use this dialog box to specify message search criteria.

### **Finder Tool**

The drag and drop tool used to select a window.

### **Hide Spy++**

Removes the main Spy++ window and leaves the **Message Search** dialog box visible on top of your other applications.

### **Handle**

The unique ID of the message to search for.

### **Type**

The message type to search for.

### **Message**

The message ID to search for.

### **Search Direction Up or Down**

The initial direction of the search.

## Messages

The Messages view displays all messages associated with windows. The window handle, message code, and message are shown.



## **Message Options: Windows**

Use this dialog box to specify message search criteria.

### **Finder Tool**

The drag and drop tool used to select a window.

### **Hide Spy++**

Removes the main Spy++ window and leaves the **Message Options** dialog box visible on top of your other applications.

### **All Windows in System**

Display messages for all windows.

### **Parent**

Display messages for the selected window and its immediate parent window.

### **Children**

Display messages for the selected window and all its child windows, including nested child windows.

### **Windows of Same Thread**

Display messages for the selected window and all other windows owned by the same thread.

### **Windows of Same Process**

Display messages for the selected window and all other windows owned by the same process.

### **Save Settings as Default**

Save the preceding settings for new message stream windows. These settings are also saved when exiting Spy++.

## **Message Options: Messages**

Use this dialog box to specify message search criteria.

### **Messages to View**

Select specific messages for viewing.

### **Message Groups**

Select message groups for viewing.

### **Save Settings as Default**

Save the current settings for later use as message search options. These settings are also saved when exiting Spy++.

## **Message Options: Output**

Use this dialog box to specify message search criteria.

### **Message Nesting Level**

Prefix nested messages with one period per level.

### **Raw Message Parameters**

Display the hexadecimal **wParam** and **lParam** values.

### **Decoded Message Parameters**

Display the results of message-specific decoding of the **wParam** and **lParam** values.

### **Raw Return Values**

Display the hexadecimal **lResult** return value.

### **Decoded Return Values**

Display the results of message-specific decoding of the **lResult** return value.

### **Message Origin Time**

The elapsed time since the Windows system was started (for posted messages only).

### **Message Mouse Position**

The screen coordinates of the mouse when the message was posted (for posted messages only).

### **Lines maximum**

Limit the number of lines that are retained in the currently selected Messages view.

### **Also Log Messages to File**

Specify an output file for the message log. This output file is written simultaneously with the message log window.

### **Save Settings as Default**

Save the preceding settings for new message stream windows. These settings are saved when you quit Spy++.

## Message Properties: General

### Window Handle

The unique ID of this window. Window handle numbers are reused; they identify a window only for the lifetime of that window. Click this value to view the properties of this window.

### Nesting Level

Depth of nesting of this message, where 0 is no nesting.

### Message

Number, status, and name of the selected windows message.

### IResult

The value of the *IResult* parameter, if any.

### wParam

The value of the *wParam* parameter, if any.

### lParam

The value of the *lParam* parameter, if any. This value is decoded if it is a pointer to a string or structure.

## **Microsoft Spy++**

The main window of Spy++, a Win32-based utility that gives you a graphical view of the system's process, threads, windows, and window messages.

## **Processes**

The Processes view displays a tree of all active processes in the system. The process ID and module name are shown. Use the Processes view if you want to examine a particular system process, which usually corresponds to an executing program. Processes are identified by module names, or they are designated “system processes.”

## **Process Properties for Windows 95: General**

### **Module Name**

The name of the module.

### **Process ID**

The Unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data.

### **Priority Base**

The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process's base priority.

### **Threads**

The number of threads currently active in this process.

## **Process Properties for Windows NT: General**

### **Module Name**

The name of the module.

### **Process ID**

The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data.

### **Priority Base**

The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process's base priority.

### **Threads**

The number of threads currently active in this process.

### **CPU Time**

The total CPU time spent on this process and its threads. Equal to User Time plus Privileged Time.

### **User Time**

The cumulative elapsed time that this process's threads have spent executing code in User Mode in non-idle threads. Applications execute in User Mode, as do subsystems such as the window manager and the graphics engine.

### **Privileged Time**

The total elapsed time this process has been running in Privileged Mode in non-idle threads. The service layer, the Executive routines, and the Kernel execute in Privileged Mode. Device drivers for most devices other than graphics adapters and printers also execute in Privileged Mode. Some work that Windows does for your application may appear in other subsystem processes in addition to Privileged Time.

### **Elapsed Time**

The total elapsed time this process has been running.

### **Priority Base**

The current base priority of this process. Threads within a process can raise and lower their own base priority relative to the process's base priority.



## **Process Properties for Windows NT: Memory**

### **Virtual Bytes**

The current size (in bytes) of the virtual address space the process is using. The use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages. However, virtual space is finite, and using too much may limit the ability of the process to load libraries.

### **Peak Virtual Bytes**

The maximum number of bytes of virtual address space the process has used at any one time.

### **Working Set**

The set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from Working Sets. If they are needed, they will be soft-faulted back into the Working Set before they leave main memory.

### **Peak Working Set**

The maximum number of pages in the working set of this process at any point in time.

### **Paged Pool Bytes**

The current amount of paged pool the process has allocated. Paged pool is a system memory area where operating system components acquire space as they accomplish their appointed tasks. Paged pool pages can be paged out to the paging file when not accessed by the system for sustained periods of time.

### **Nonpaged Pool Bytes**

The current number of bytes in the nonpaged pool allocated by the process. The nonpaged pool is a system memory area where space is acquired by operating system components as they accomplish their appointed tasks. Nonpaged pool pages cannot be paged out to the paging file; they remain in main memory as long as they are allocated.

### **Private Bytes**

The current number of bytes this process has allocated that cannot be shared with other processes.

### **Free Bytes**

The total unused virtual address space of this process.

### **Reserved Bytes**

The total amount of virtual memory reserved for future use by this process.

### **Free Image Bytes**

The amount of virtual address space that is not in use or reserved by images within this process.

### **Reserved Image Bytes**

The sum of all virtual memory reserved by images run within this process.

## **Process Properties for Windows NT: Page File**

### **Page File Bytes**

The current number of pages that this process is using in the paging file. The paging file stores pages of data used by the process but not contained in other files. The paging file is used by all processes, and lack of space in the paging file can cause errors while other processes are running.

### **Peak Page File Bytes**

The maximum number of pages that this process has used in the paging file.

### **Page Faults**

The number of Page Faults by the threads executing in this process. A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. Thus, the page will not be retrieved from disk if it is on the standby list and hence already in main memory, or if it is being used by another process with which the page is shared.

## **Process Properties for Windows NT: Space**

### **Show for space marked as**

Use this listbox to select the category of space (image, mapped, reserved, or unassigned).

### **Executable Bytes**

For the selected category, the sum of all the address space that this process is using. Executable memory is memory that can be executed by programs, but may not be read or written.

### **Exec-Read-Only Bytes**

For the selected category, the sum of all the address space in use with read-only properties that this process is using. Exec-read-only memory is memory that can be executed as well as read.

### **Exec-Read-Write Bytes**

For the selected category, the sum of all the address space in use with read-write properties that this process is using. Exec-read-write memory is memory that can be executed by programs as well as read and modified.

### **Exec-Write Copy Bytes**

For the selected category, the sum of all the address space that can be executed by programs as well as read and written. This type of protection is used when memory needs to be shared between processes. If the sharing processes only read the memory, then they will all use the same memory. If a sharing process desires write access, then a copy of this memory will be made for the process.

### **No-Access Bytes**

For the selected category, the sum of all the address space that prevents a process from using it. An access violation is generated if writing or reading is attempted.

### **Read-Only Bytes**

For the selected category, the sum of all the address space that can be executed as well as read.

### **Read-Write Bytes**

For the selected category, the sum of all the address space that allows reading and writing.

### **Write-Copy Bytes**

For the selected category, the sum of all the address space that allows memory sharing for reading but not for writing. When processes are reading this memory, they can share the same memory. However, when a sharing process wants to have read/write access to this shared memory, a copy of that memory is made for writing.

## **Threads**

The Threads view displays a flat listing of all active threads associated with windows. The Thread ID and module name are shown. Processes are not included.

## **Thread Properties for Windows 95: General**

### **Module Name**

The name of the module.

### **Thread ID**

The unique ID of this thread. Note that thread ID numbers are reused; they identify a thread only for the lifetime of that thread.

### **Process ID**

The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data. Click on this value to view the properties of the process ID.

### **Current Priority**

The current dynamic priority of this thread. Threads within a process can raise and lower their own base priority relative to the base priority of the process.

### **Base Priority**

The current base priority of this thread.

## **Thread Properties for Windows NT: General**

### **Module Name**

The name of the module.

### **Thread ID**

The unique ID of this thread. Note that thread ID numbers are reused; they identify a thread only for the lifetime of that thread.

### **Process ID**

The unique ID of this process. Process ID numbers are reused, so they identify a process only for the lifetime of that process. The Process object type is created when a program is run. All the threads in a process share the same address space and have access to the same data. Click on this value to view the properties of the process ID.

### **Thread State**

The current state of the thread. A Running thread is using a processor; a Standby thread is about to use one. A Ready Thread is waiting to use a processor because one is not free. A thread in Transition is waiting for a resource to execute, such as waiting for its execution stack to be paged in from disk. A Waiting thread does not need the processor because it is waiting for a peripheral operation to complete or a resource to become free.

### **Wait Reason**

Applicable only when the thread is in the Wait state. Event Pairs are used to communicate with protected subsystems.

### **CPU Time**

The total CPU time spent on this process and its threads. Equal to User Time plus Privileged Time.

### **User Time**

The total elapsed time that this thread has spent executing code in User Mode. Applications execute in User Mode, as do subsystems like the window manager and the graphics engine.

### **Privileged Time**

The total elapsed time that this thread has spent executing code in Privileged Mode. When a Windows system service is called, the service will often run in Privileged Mode to gain access to system-private data. Such data is protected from access by threads executing in User Mode. Calls to the system may be explicit, or they may be implicit, such as when a page fault or an interrupt occurs.

### **Elapsed Time**

The total elapsed time (in seconds) this thread has been running.

### **Current Priority**

The current dynamic priority of this thread. Threads within a process can raise and lower their own base priority relative to the base priority of the process.

### **Base Priority**

The current base priority of this thread.

### **Start Address**

The starting virtual address for this thread.

**User PC**

The user program counter for the thread.

**Context Switches**

The number of switches from one thread to another. Thread switches can occur either inside a single process or across processes. A thread switch may be caused by one thread asking another for information, or by a thread being preempted when a higher priority thread becomes ready to run.

## **Windows**

The Windows view displays a tree of all active windows and controls in the system. The window handle and class name are shown. The current desktop window is at the top of the tree. All other windows are children of the desktop, and are listed according to the standard window hierarchy, with sibling windows ordered by depth.



## **Window Properties: General**

### **Window Caption**

The text in the window caption.

### **Window Handle**

The unique ID of this window.

### **Window Proc**

The virtual address of the window procedure function for this window. This field also indicates whether this window is a Unicode window, and whether it is subclassed.

### **Rectangle**

The bounding rectangle for the window. The size of the rectangle is also displayed. Units are pixels in screen coordinates.

### **Restored Rect**

The bounding rectangle for the restored window. The size of the rectangle is also displayed. Restored Rect will differ from Rectangle only when the window is maximized or minimized. Units are pixels in screen coordinates.

### **Client Rect**

The bounding rectangle for the window client area. The size of the rectangle is also displayed. Units are pixels relative to the top left of the window.

### **Instance Handle**

The instance handle of the application. Instance handles are not unique.

### **Menu Handle**

If the window being displayed is a child window, the Control ID label is displayed. Control ID is an integer that identifies this child window's control ID. If the window being displayed is not a child window, the Menu Handle label is displayed. Menu Handle is an integer that identifies the handle of the menu associated with this window.

### **User Data**

Application-specific data that is attached to this window structure.

### **Window Bytes**

The number of extra bytes associated with this window. The meaning of these bytes is determined by the application. Expand the list box to see the byte values in DWORD format.

## **Window Properties: Styles**

### **Window Styles**

The combination of window style codes.

### **Extended Styles**

The combination of extended window style codes.

## **Window Properties: Windows**

### **Next Window**

The handle of the next sibling window in the same sequence shown in the window tree view (“none” if there is no next window). Click on this entry to view the properties of the next window.

### **Previous Window**

The handle of the previous sibling window in the same sequence shown in the window tree view (“none” if there is no previous window). Click on this entry to view the properties of the previous window.

### **Parent Window**

The handle of this window’s parent window (“none” if there is no parent). Click on this entry to view the properties of the parent window.

### **First Child**

The handle of this window’s first child window, in the sequence shown in the window tree view (“none” if there are no child windows). Click on this value to view the properties of the first child window.

### **Owner Window**

The handle of this window’s owner window. An application’s main window typically owns system-modal dialog windows, for example (“none” if there is no owner). Click on this entry to view the properties of the owner window.

## **Window Properties: Class**

### **Class Name**

The name (or ordinal number) of this window class.

### **Class Styles**

A combination of class style codes.

### **Class Bytes**

Application-specific data associated with this window class.

### **Class Atom**

The atom for the class returned by the **RegisterClass** call.

### **Instance Handle**

The instance handle of the module that registered the class. Instance handles are not unique.

### **Window Bytes**

The number of extra bytes associated with each window of this class. The meaning of these bytes is determined by the application. Expand the list box to see the byte values in DWORD format.

### **Window Proc**

The current address of the **WndProc** function for windows of this class. This differs from **Window Proc** on the **General** tab if the window is subclassed.

### **Menu Name**

The name of the main menu that is associated with windows of this class (“none” if there is no menu).

### **Icon Handle**

The handle for the icon that is associated with windows of this class (“none” if there is no icon).

### **Cursor Handle**

The handle for the cursor that is associated with windows of this class (“none” if there is no cursor).

### **Bkgnd Brush**

The handle for the background brush that is associated with windows of this class, or one of the predefined COLOR\_\* colors for painting the window background (“none” if there is no brush).

## **Window Properties: Process**

### **Process ID**

The ID of the process that owns the thread that created this window. Click on this value to view the properties of this process.

### **Thread ID**

The ID of the thread that created this window. Click on this value to view the properties of this thread.

## **Window Search**

Use this dialog box to specify window search criteria.

### **Finder Tool**

The drag and drop tool used to select a window.

### **Hide Spy++**

Removes the main Spy++ window and leaves the **Window Search** dialog box visible on top of your other applications.

### **Handle**

The unique ID of the window to search for.

### **Caption**

The text in the window caption to search for.

### **Class**

The class of the window to search for.

### **Search Direction Up or Down**

The initial direction of the search.

