



WindowsSCOPE Cyber Forensics - Ultimate Quick Start Guide

<http://www.windowsscope.com/>

Table of Contents

Overview.....	1
Getting Started.....	1
Structure Map.....	2
OS Structures.....	3
Processes.....	3
Drivers.....	4
Summary of System Activity.....	5
Investigate.....	5
Raw.....	5
Disassembled.....	6
Graph.....	7
Compare.....	9
Configure View.....	9
Licensing and Configuration.....	10
Importing Memory Snapshots.....	10
Glossary.....	11

Overview

The WindowsSCOPE application provides many useful Microsoft Windows kernel analysis and memory forensics features. It has the ability to take and build repositories of snapshots of physical and virtual memory (“fetching”) and furnishes detailed operating system information from a memory snapshot. In addition, WindowsSCOPE users can review detailed information about each running process and driver, such as open files, network sockets, and registry keys and the raw code, strings, and symbols from each of the process's modules. The code can be disassembled, and the disassembled instructions can be graphed in a program flow diagram for an analysis aid. Components of each snapshot can be compared against the same components of other snapshots. (For application notes, please review the tutorial videos available on the WindowsSCOPE website.)

Getting Started

This section will provide step-by-step instructions for exploring WindowsSCOPE through one of the example snapshots provided with the software if the “Sample Repository” option was selected during installation. If you haven't already installed your license, you'll be prompted with the dialog in Figure 1. If you haven't already obtained a license file, you will need to get one by either purchasing a license or registering for a trial license according to step 1 in the dialog. Once you have completed a purchase order or registered for a trial you will be able to follow step 2 to request a license file. After the license file has been requested you will receive an email with the license file attached. Now you can save a copy of the license file and install it in WindowsSCOPE by clicking the



Figure 1: License installation.

Install License button in the license manager. Once the license file is installed you can click the Close button in the license manager and continue using WindowsSCOPE.

WindowsSCOPE operates on projects; the first thing you'll want to do when you start is open a project, either by choosing an existing project or by creating a new one. Start by opening the included sample project: choose Project»Open... from the menu, select “Sample Project” from the list, and click the Open button.



Figure 2: Main tab bar and Structure Map tree pane.

The primary means of controlling WindowsSCOPE is the main tab bar. By default, a project will open in the Structure Map tab, which will display the tree shown in Figure 2. This tab allows you to explore the contents of the memory snapshot. Data is organized into four sections: Operating System Structures, Processes, Drivers, and a Summary of System Activity. The Investigate tab provides tools for

analyzing the text of modules in memory, including disassembly and graphing. The Compare tab compares a table from the Structure Map tab across two memory snapshots.

Click the ▼ next to “Select Snapshot” and choose “Snapshot 1” from the list. The tree now shows some basic information about the snapshot and the Memory View node. Expand this node by clicking the ⊕ symbol beside it; the nodes it contains are described in more detail in the Structure Map section of this guide.

If you would like to explore the current machine's virtual memory, rather than the example snapshots provided, click on the Fetch... button below the Structure Map tree. A dialog will appear; click on the Fetch button to take the snapshot and wait for it to finish. (This could take several minutes; the exact time depends on the number of active processes and drivers on your machine, as well as its speed, physical and virtual memory sizes, and other factors.) When the process is complete, you may begin using WindowsSCOPE to explore your machine's virtual memory. WindowsSCOPE Ultimate also allows you to import snapshots from other computers to your repository. Snapshots captured with the WindowsSCOPE Phantom Probe USB dongle and raw memory dumps are supported. Details on importing memory snapshots from other computers will be covered in the next section, “Importing Memory Snapshots”.

Below the tree is the Fetch/Import button. As described in the previous section, clicking this button yields the Fetch Memory Snapshot dialog from Figure 3. Selecting a memory option and clicking this button will begin fetching or importing a memory snapshot. This may take several minutes. When the process finishes, a dialog will request a name for the snapshot, and it will be added to the currently open project and stored in the current repository.

Importing Memory Snapshots

This section will discuss how to capture memory snapshots from other computers and how to import those memory snapshots into the repository on your computer where WindowsSCOPE is installed. WindowsSCOPE Ultimate can import any memory dump in raw format, as well as memory dumps taken with Phantom Probe from a USB drive.

To create a memory snapshot with Phantom Probe, simply plug the USB drive into the computer being investigated, and run the file “PhantomProbe.bat” (Note: you MUST run this as administrator, either by right clicking and selecting “Run as administrator”, or running from an elevated command prompt). Once running you will see progress information displayed

to the console until the memory fetch completes. Please note that the time required to fetch snapshots with Phantom Probe can vary significantly, depending on factors such as number of running processes, total amount of memory, and resource utilization.

When the memory fetch is completed, be sure to use the “Safely Remove Hardware” feature in Windows to protect your USB dongle. After removing the USB drive, bring it to your analysis machine where WindowsSCOPE is installed, and plug it in. Be sure to take note of the drive letter that gets assigned to the USB drive, as you will need to locate the memory snapshot to import it. Then, in WindowsSCOPE, either open an existing project or create a new project to import the memory snapshot into. Click on the “Fetch” button in the lower left part of the screen. Select the second option, “Import Phantom Probe Memory Snapshot from USB Dongle”, and click the “Browse” button. Now you will need to navigate to your USB drive and locate the memory snapshot. Once in the root directory of the USB drive, look for folders that have a name starting with “O1_F_” - each of these folders contains a memory snapshot. The rest of the file name is a time stamp that indicates when the memory snapshot was taken. Open the folder for the memory snapshot you want to import, and then select the file named “snapshot.brm” inside the folder.

At this point it will take several minutes for files to copy from the USB drive to your local repository. When finished you have the option of giving the snapshot a name to help you identify it. Once imported, it will take a moment for the contents of the snapshot to display, and then you will be able to navigate the memory snapshot.

To import a raw format memory dump, first open the WindowsSCOPE project you want to import the snapshot to. Then click the “Fetch” button in the lower left part of the screen. Select the third memory fetch option, “Import Raw Format Memory Dump File”. Then you will be prompted to select the memory dump file that you wish to import.

After selecting the memory dump file, click the “Fetch” button to begin the import. After several minutes you will be prompted to name the memory snapshot. Choose a name that will help you identify the snapshot. In a few moments the contents of the memory snapshot will be displayed, and you will be able to navigate through its contents.

Structure Map

Selecting the Structure Map tab reveals two components: on the left, a tree for navigating the snapshot (Figure 4); on the right, a pane for examining portions of the structure. The tree lists information about the snapshot, such as the computer's name, operating system, memory size and architecture and the time the snapshot was taken, and is navigated by clicking the \oplus and \ominus symbols next to tree nodes, indicated by a  (or a  when expanded).

Following the information, the Memory View node contains four sub-nodes: OS Structure, Processes, Drivers, and Summary of System Activity. The following sections of this guide describe the function of each node. Selecting most nodes displays two tables in the pane on the right. The top table is filled with details of the selected node; the bottom table lists other snapshots containing the same node for comparison—described later in this guide; see the Compare section for more information on these features.

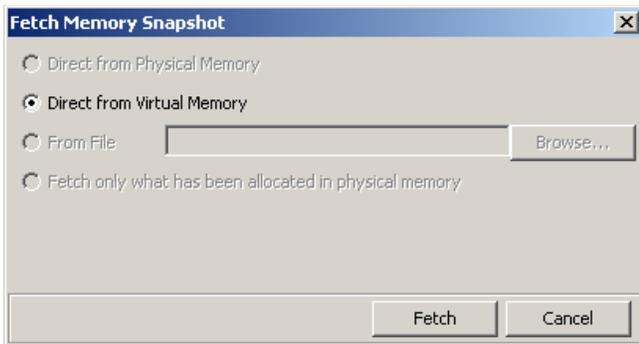


Figure 3: The snapshot fetching dialog

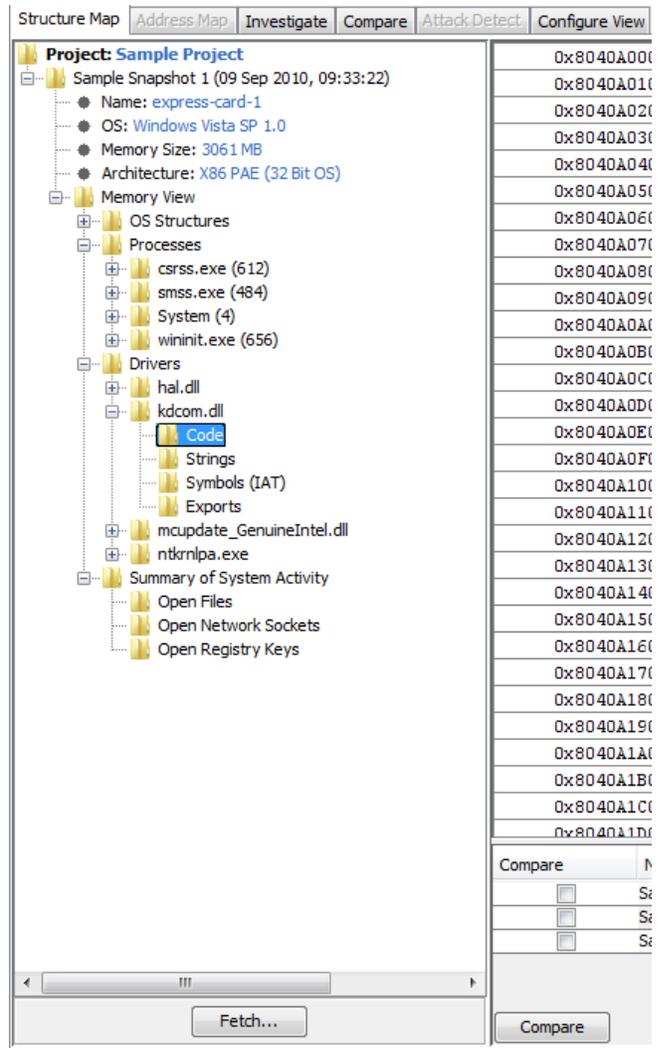


Figure 4: The tree pane of the Structure Map tab

OS Structures

The OS Structures node contains four nodes of its own: SSDT, IDT, a Page Directory Pointer list, and Process Table.

- **SSDT** – The System Service Descriptor Table is a table of pointers to functions exported by the kernel, used for completing system calls at the kernel level. Selecting this node displays a five-column table. The Hooked column indicates whether each entry was hooked at the time of the snapshot. (This feature is enabled by analysis plugins and is not available in all versions.) The preceding four columns indicate the entry number, memory address, module, and function name (if available) for the target of each system call.
- **IDT** – The Interrupt Descriptor Table is a structure used by Windows to support handling processor exceptions and interrupts. It contains pointers to the interrupt handlers for each type of interrupt or exception. Selecting this node displays a six-column table. Each of the 256 interrupt entries lists whether it has been hooked, its type (TASK, INTERRUPT or TRAP), and the address, module, and function name (if available) of the target.
- **Page Directory** – This node is organized in a hierarchy of drop-down boxes. For PAE (Physical Address Extension) systems, the top level node is a list of Page Directory Pointers by process name. When the Page Directory Pointer is selected, the next level down is a list of Page Directories, of which each process has four, numbered 0–3. For non-PAE systems, the top level node is a list of Page Directories by process name. Once a Page Directory is selected for either type of system (PAE or non-PAE), the next level down in the hierarchy is a list of Page Tables—numbered 0–511 for PAE systems and 0–1023 for non-PAE systems. The next level is a list of 64-entry ranges of Page Table

Entries. (If blank entries have been hidden in the Configure View tab, only populated ranges appear here.) Selecting one of these options reveals its range of entries as nodes; selecting one of these nodes generates a diagram in the pane on the right displaying the entry's page frame number, flags, and content, as shown in Figure 5.

- **Process Table** – Selecting the process table displays a table with three columns: Process Name, Hidden, and PID, which represent the name of each process, whether or not the process was flagged as hidden, and the process identification number, as reported by Windows.

Processes

- Expanding the *Processes* node reveals a list of processes running at the time of the snapshot, each containing the following nodes:
- **Modules** – This node contains a set of nodes listing each module involved in the current process. Each module node contains three nodes:
 - **Code** – Selecting this node displays the content of the module in a list of sixteen-byte words by address, showing both the hexadecimal values of each byte and the associated printable character.
 - **Strings** – Selecting this node displays a four-column table listing each string in the module, the offset of the string in the module, the type of the string (ANSI or Unicode), and the length of the string.

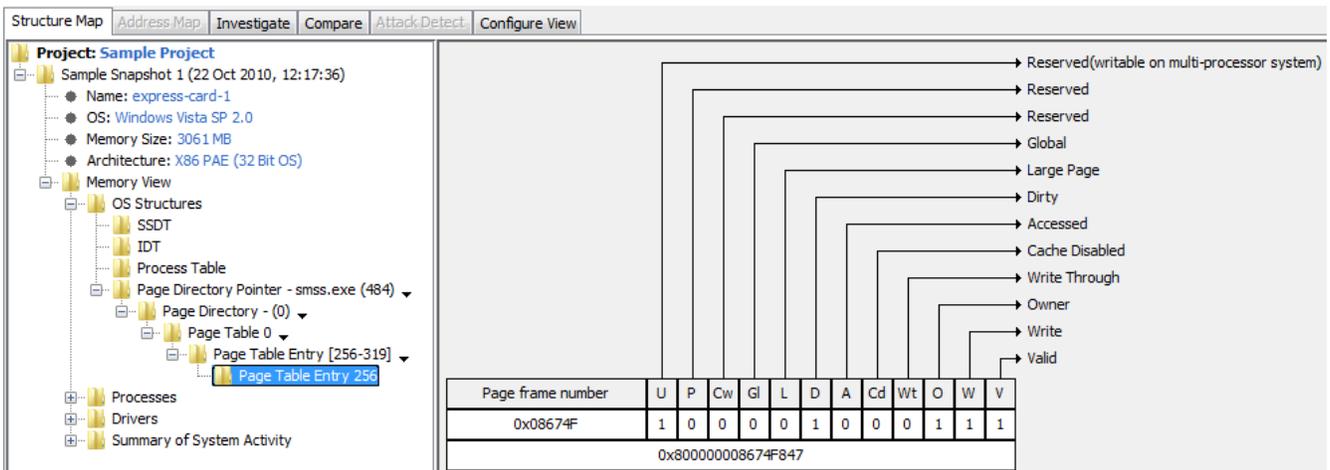


Figure 5: View of a page table entry from the sample project

- **Symbols (IAT)** – A list of symbols is retrieved from the Import Address Table, which is a table of pointers to functions that have been imported from external DLL files. Selecting this node displays a four-column table listing each symbol in the module, the address of the symbol in the module, the address of the IAT entry, and the module from which the symbol was imported.
 - **Exports** – A list of exports is retrieved from the Export Address Table, which is a table containing a list of functions and variables that are made available for other programs and DLLs to use. Selecting this node displays a four-column table listing the ordinal number for each export, its name, the address of the exported item, and the address of the export table entry.
- **Open Files** – Selecting this node displays a three-column table listing the name of each open file, the handle's owning process, and the access to the file.
- **Open Network Sockets** – Selecting this node displays a five-column table listing the source IP and port, destination IP and port, type (TCP or UDP), owning process, and connection state of each socket connection.
- **Open Registry Keys** – Selecting this node displays a two-column table listing the name of each key and the handle's owning process.
- **Threads** – Selecting this node displays a three-column table listing the identification number of each thread belonging to the process, the priority of the thread, and the thread's owning process.

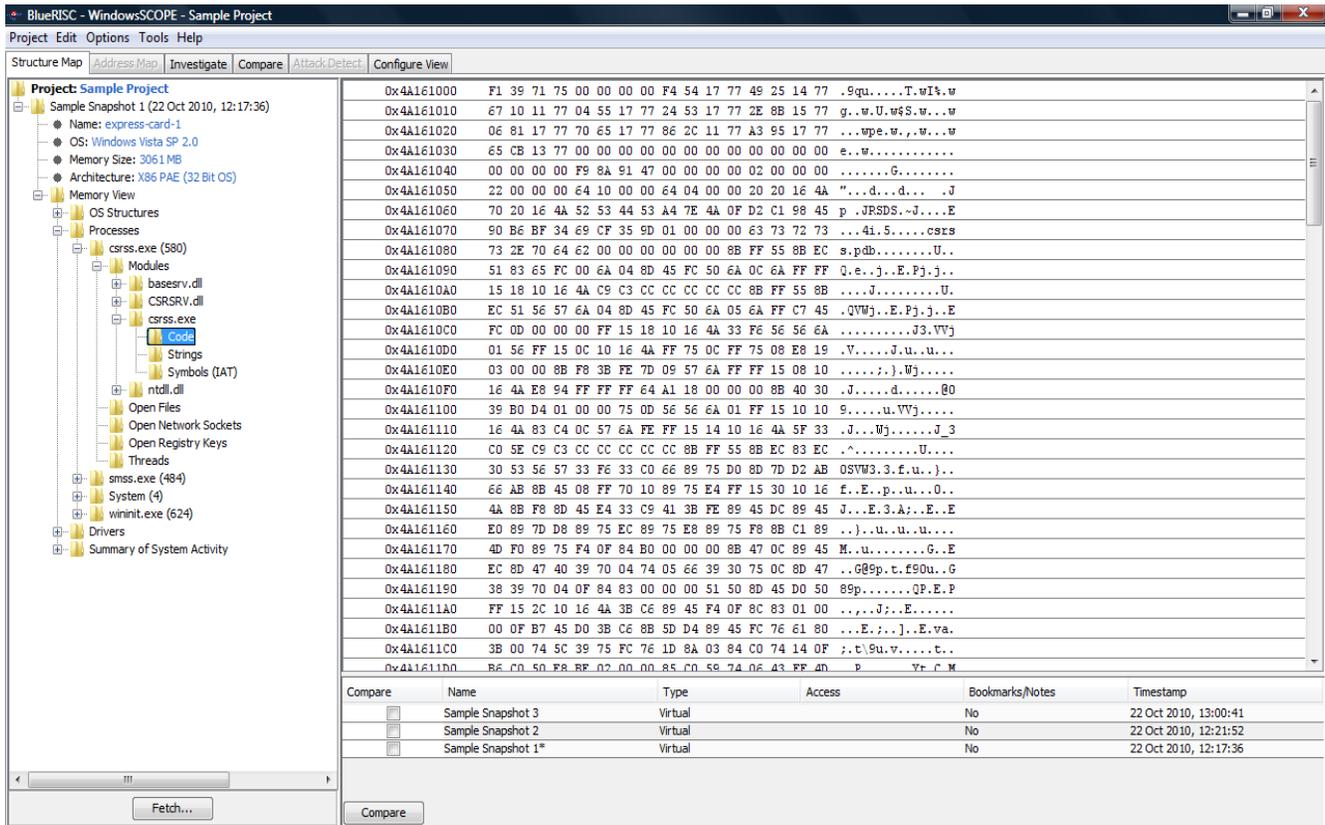


Figure 6: Selecting the Code node displays the content of the module in the pane on the right. The information displayed in hexadecimal code is identical to the set of characters to the right: each of the sixteen bytes is presented in the same order. Non-standard and control characters are represented by a period. Note that the entire raw text can be compared byte-for-byte with the same module from another snapshot using the Compare feature below the table, discussed later in this guide.

Drivers

In Windows, a driver is a program designed to allow high-level software to interact with hardware. Expanding the Drivers node reveals a list of drivers loaded at the time of the snapshot, each containing the following nodes:

- **Code** – Selecting this node displays the content of the driver in a list of sixteen-byte words by address, showing both the hexadecimal values of each byte and the associated printable character.
- **Strings** – Selecting this node displays a four-column table listing each string in the driver, the offset of the string in the driver, the type of the string (ANSI or Unicode), and the length of the string.
- **Symbols (IAT)** – A list of symbols is retrieved from the Import Address Table, which is a table of pointers to functions that have been imported from external DLL files. Selecting this node displays a four-column table listing each symbol in the module, the address of the symbol in the module, the address of the IAT entry, and the module from which the symbol was imported.
- **Exports** – A list of exports is retrieved from the Export Address Table, which is a table containing a list of functions and variables that are made available for other programs and DLLs to use. Selecting this node displays a four-column table

listing the ordinal number for each export, its name, the address of the exported item, and the address of the export table entry.

Summary of System Activity

Expanding the *Summary of System Activity* node reveals the following three nodes:

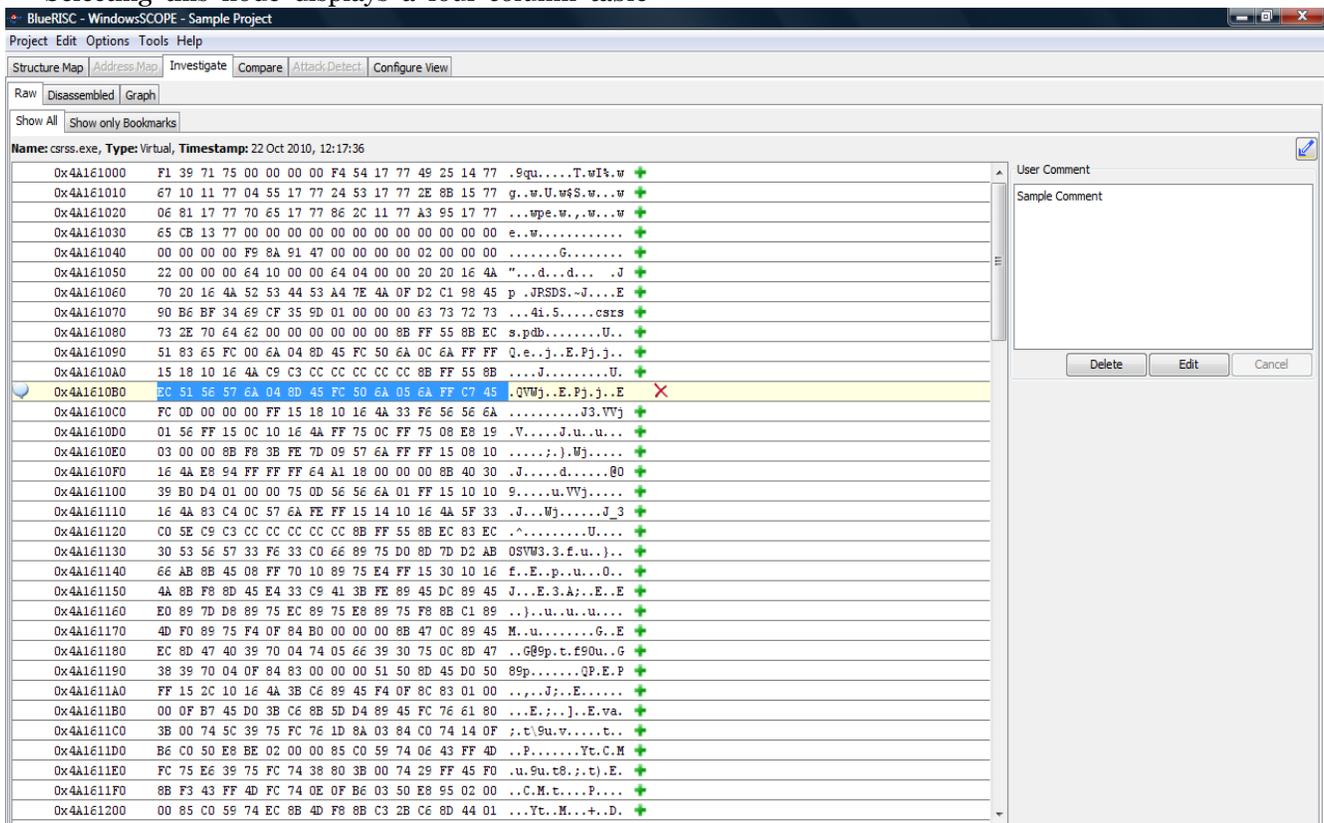
- **Open Files** – Selecting this node concatenates the Open Files list from each process into one table.
- **Open Network Sockets** – Selecting this node concatenates the Open Network Sockets list from each process into one table.
- **Open Registry Keys** – Selecting this node concatenates the Open Registry Keys list from each process into one table.

Investigate

Selecting the Investigate tab reveals, below it, three additional tabs: Raw, Disassembled, and Graph. Its content is linked from the last selected Code node in the Structure Map tab.

Raw

The Raw tab is selected by default. It contains two additional tabs, Show All and Show only Bookmarks.



Under the Show All tab is a table nearly identical to the Text table from the Structure Map. However, at the end of each line, a **+** enables the user to add a comment to the line. Once a comment is added, the **+** disappears, and an **X** replaces it, which allows the user to delete the comment. Commented lines are bookmarked; a **📌** appears at the beginning of the line to indicate this. Switching to the Show only Bookmarks tab hides all lines without comments. To the right of the list in both tabs is a text box containing the comment for the currently selected line and three buttons beneath it for deleting, editing, and canceling changes to the comment. Additionally, right-clicking a line offers the option to jump to the same portion of code in the Disassembled tab. Selecting **📌** highlights the portions of the code which were in physical memory at the time of the snapshot with a light green background.

Disassembled

The Disassembled tab sports the same comment system as the Raw tab, including the two Show All and Show only Bookmarks tabs and the User Comment box to the right. However, each line is an assembly instruction parsed from the raw code. After the address, instead of sixteen-byte words, comes the offset of each instruction in the section, the opcode and operands of the instruction, and the original hexadecimal code of the instruction from the raw code. Right-clicking on a line, like the similar function in the Raw tab, offers an option to jump to the same instruction in the Graph tab. Selecting **📌** highlights the portions of the code which were in physical memory at the time of the snapshot with a light green background.

Right-clicking also offers the option “Graph from Here: Workpad”. This will graph instructions and procedures that are near the selected instruction. This is useful when you are only interested in program behavior near the selected instruction. The graphing process will be much faster and the resulting graph will be much smaller and easier to navigate. More details on the graph workpad can be found in the Graph section.

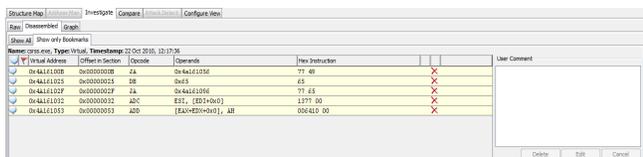


Figure 7: The disassembled view when only showing instructions containing comments.

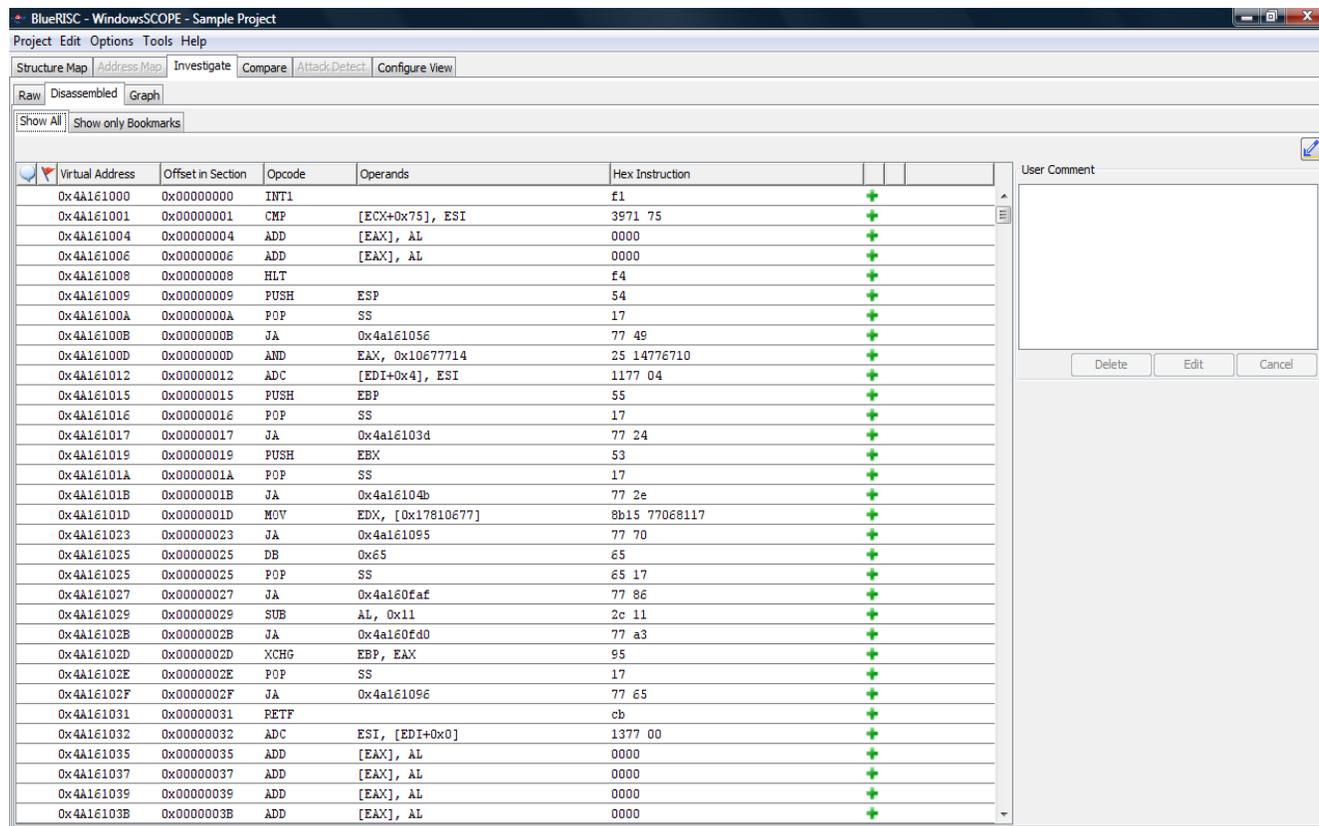


Figure 8: Disassembled view when investigating a Code node from the Structure Map tree.

Graph

After disassembling the raw code, switching to the Graph tab generates a visual representation of the program flow for the selected module on the assembly level. The processing required for generating the graph typically will not take more than 1 or 2 minutes for moderately sized modules or drivers. Some large modules and drivers can take longer, so if you don't need the entire graph you should consider using the workpad to view the parts of the program you're interested in. The workpad graph sub-tab is described later in this section.

Once the graph has been generated you will be brought to the Full View sub-tab where three panes are visible. On the left is a list of procedures; in the center is a procedure-level view, showing the program flow between procedures; and on the right, once a procedure is selected, is a detailed view of the flow of instructions and basic blocks within the procedure.

Selecting a procedure in the list, in the main diagram, or in the right-hand procedure view will update the procedure view with the details of the new procedure. Choosing the  in the top right of both the main diagram and the procedure view will expand the respective view over both of them; choosing  will return the view to normal. Additionally, the procedure view has a ; clicking it will return the view to the previous procedure. Example graph views

are displayed in Figures 9 and 10 on the following pages.

When you switch to the Workpad sub-tab there is a drop down list that allows you to load a saved graph. If you don't have one yet you can generate a workpad graph by going to the disassembled view and right-clicking an instruction you want to use as the start point for a workpad graph. After a few seconds the graph will be generated and shown in the Workpad sub-tab of the Graph view. Workpad graphs will typically be much smaller and easier to navigate and will take much less time to create. If you create a workpad graph that you want to save for later use you can click the Save button at the top of the view, which will prompt you to provide a name for the graph. After you have saved workpad graphs you can get back to them by going to the graph tab for the module or driver the workpad graphs were saved for, and then switching to the Workpad sub-tab. The saved graphs will be available in the dropdown list at the top of the workpad view.

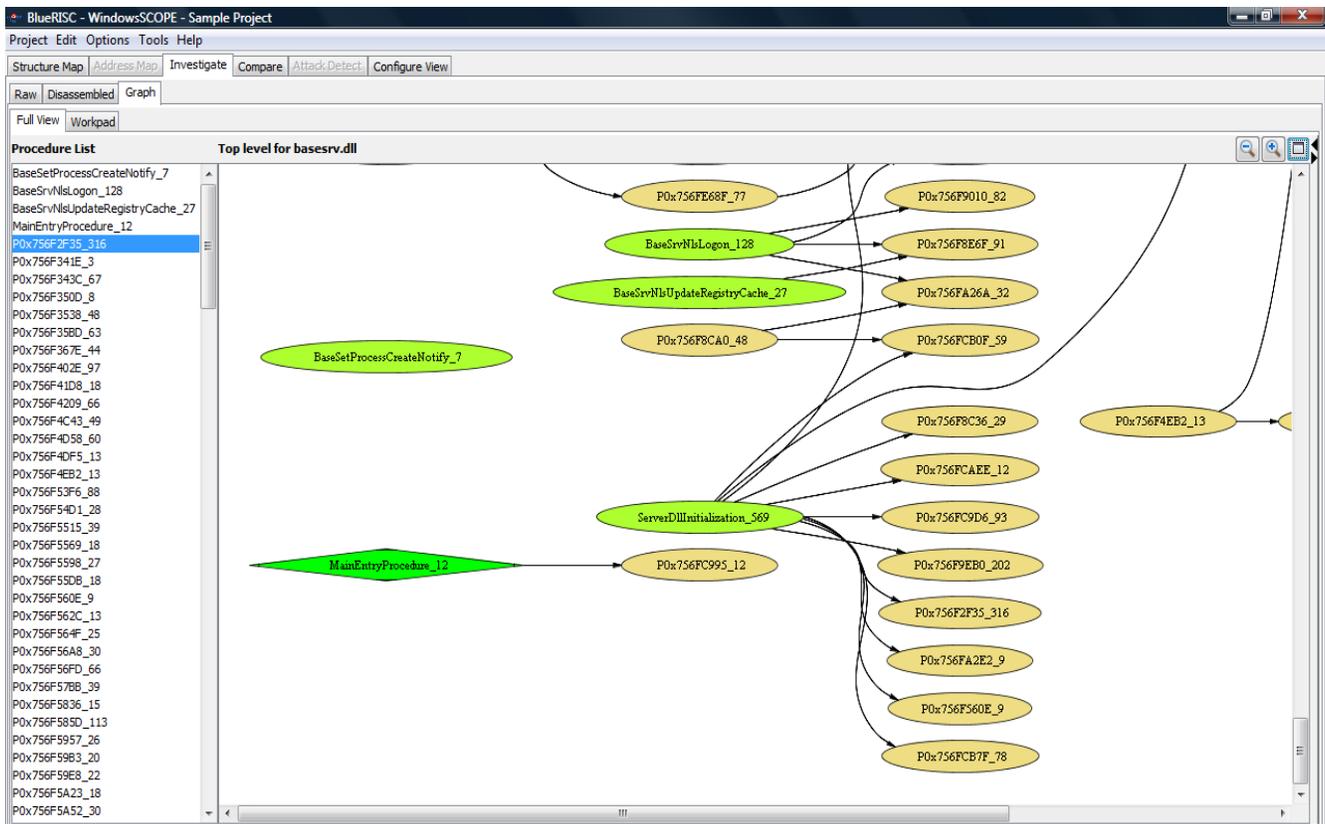


Figure 9: Initial view of a graph. Each of the oval-shaped nodes may be selected for inspection and has an equivalent entry in the list on the left.

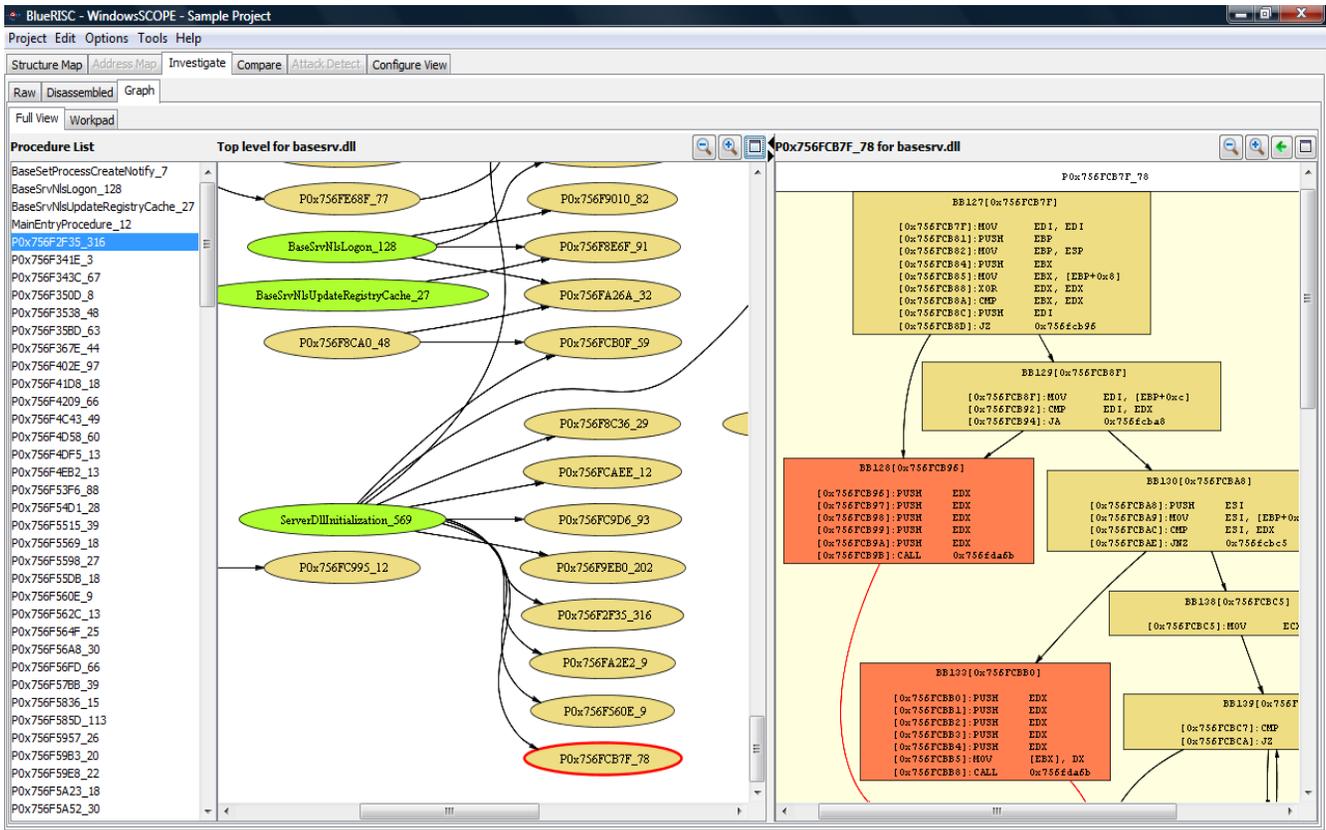


Figure 10: When a node is selected for inspection, the code is displayed on the right in a flow diagram.

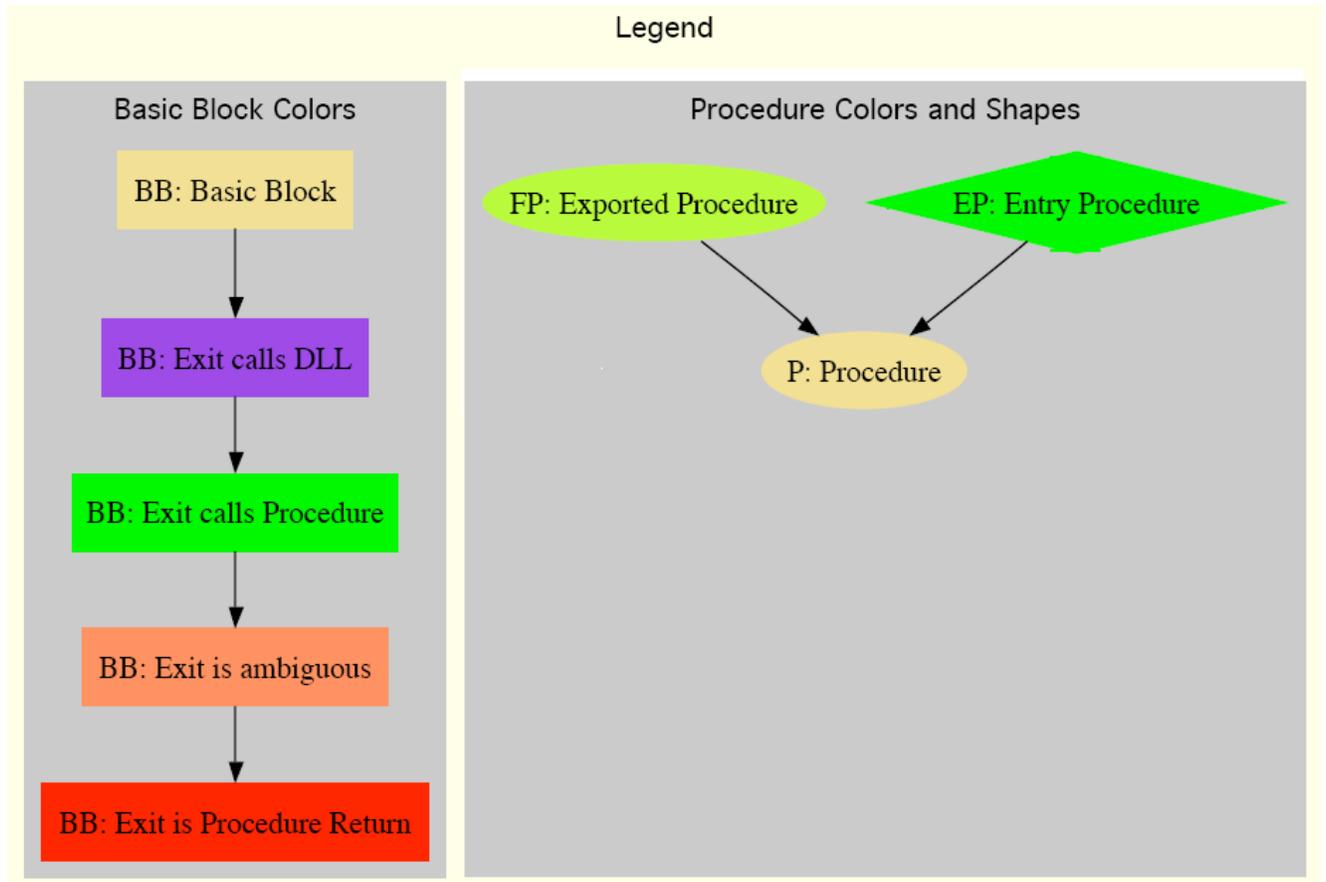


Figure 11: Legend for interpreting control flow graphs

Compare

Selecting the Compare tab will reveal a blank viewport unless a previous comparison has been performed. To perform a comparison, switch back to the Structure Map tab, select a node with table data, and check the boxes next to two snapshots in the table below the data, as shown in Figure 12. This table lists all snapshots in the current project containing the selected node.

Clicking the Compare button switches the view to the Compare tab, which now shows the configuration options for the comparison. There are two sets of settings in the compare view – one for each snapshot that is being compared. For each snapshot you can set the position in the table to start the comparison, as well as the number of rows to compare. By default there will be settings entered that will perform a quick comparison, starting at the beginning of each table. Note that increasing the comparison lengths can dramatically increase the time required to perform the comparison, especially when there are very few similarities between the tables being compared. You can see the comparison configuration options in Figure 13.

avgrsx.exe	1436	Not Checked
MotoConnectService.exe	2064	Not Checked

Compare	Name	Type	Access	Bookmarks/Notes	Timestamp
<input type="checkbox"/>	Sample Snapshot 3	Virtual	No		22 Oct 2010, 13:00:41
<input type="checkbox"/>	Sample Snapshot 2	Virtual	No		22 Oct 2010, 12:21:52
<input type="checkbox"/>	Sample Snapshot 1*	Virtual	No		22 Oct 2010, 12:17:36

Compare

Figure 12: The Compare table in the Structure Map tab.

BlueRISC - WindowsSCOPE - Sample Project

Project Edit Options Tools Help

Structure Map Address Map Investigate Compare Attack Detect Configure View

Sample Snapshot 2
Timestamp: 22 Oct 2010, 12:21:52
Name: Process Table
Start Row Number: 0
Comparison Length: 95

Sample Snapshot 1
Timestamp: 22 Oct 2010, 12:17:36
Name: Process Table
Start Row Number: 0
Comparison Length: 102

Compare Statistics
Number of Lines Changed: 0
Number of Lines Removed: 1
Number of Lines Inserted: 4

Process Name	PID	Hidden
dmn.exe	3040	Not Checked
explorer.exe	2172	Not Checked
MSASCui.exe	3608	Not Checked
Apoin.exe	3648	Not Checked
hkcmd.exe	4104	Not Checked
igfxpers.exe	4160	Not Checked
igfxsrvc.exe	4240	Not Checked
IAAnotif.exe	4252	Not Checked
WLTRAY.EXE	4356	Not Checked
ApMsgFwd.exe	4544	Not Checked
ApntEx.exe	4568	Not Checked
hidfind.exe	4576	Not Checked
PCMSservice.exe	4692	Not Checked
sprtcmd.exe	4700	Not Checked
avgtRAY.exe	4724	Not Checked
AdobeARM.exe	4788	Not Checked
sttRAY.exe	4808	Not Checked
jusched.exe	4860	Not Checked
mswinext.exe	4904	Not Checked
daemon.exe	4952	Not Checked
ehtray.exe	4968	Not Checked
ehmsas.exe	4988	Not Checked
DLG.exe	5000	Not Checked
wapnscfg.exe	5344	Not Checked
firefox.exe	4196	Not Checked
plugin-container.exe	4648	Not Checked
thunderbird.exe	2076	Not Checked
cedt.exe	5632	Not Checked
avgrsx.exe	5596	Not Checked
swriter.exe	5308	Not Checked
soffice.exe	5448	Not Checked
soffice.bin	1216	Not Checked
googletalkplugin.exe	3220	Not Checked
taskeng.exe	2676	Not Checked
wuauclt.exe	5824	Not Checked
cmd.exe	5712	Not Checked
mspdsrv.exe	2980	Not Checked
java.exe	3888	Not Checked
WerFault.exe	4232	Not Checked

Process Name	PID	Hidden
dmn.exe	3040	Not Checked
explorer.exe	2172	Not Checked
MSASCui.exe	3608	Not Checked
Apoin.exe	3648	Not Checked
hkcmd.exe	4104	Not Checked
igfxpers.exe	4160	Not Checked
igfxsrvc.exe	4240	Not Checked
IAAnotif.exe	4252	Not Checked
WLTRAY.EXE	4356	Not Checked
ApMsgFwd.exe	4544	Not Checked
ApntEx.exe	4568	Not Checked
hidfind.exe	4576	Not Checked
PCMSservice.exe	4692	Not Checked
sprtcmd.exe	4700	Not Checked
avgtRAY.exe	4724	Not Checked
AdobeARM.exe	4788	Not Checked
sttRAY.exe	4808	Not Checked
jusched.exe	4860	Not Checked
mswinext.exe	4904	Not Checked
daemon.exe	4952	Not Checked
ehtray.exe	4968	Not Checked
ehmsas.exe	4988	Not Checked
DLG.exe	5000	Not Checked
wapnscfg.exe	5344	Not Checked
firefox.exe	4196	Not Checked
plugin-container.exe	4648	Not Checked
thunderbird.exe	2076	Not Checked
mspaint.exe	5644	Not Checked
XWin.exe	5108	Not Checked
xterm.exe	1880	Not Checked
bash.exe	4836	Not Checked
cedt.exe	5632	Not Checked
impact.exe	4320	Not Checked
_impact.exe	5568	Not Checked
impactosth.exe	2080	Not Checked
avgrsx.exe	5596	Not Checked
Scribus.exe	1172	Not Checked
swriter.exe	5308	Not Checked
soffice.exe	5448	Not Checked

Figure 13: View of the Compare tab for the process tables of the sample project's Snapshot 1 and Snapshot 2.

Once the comparison has been configured you can start the compare by clicking the Compare button. When the comparison is finished the comparison results will be shown as in Figure 13. Text above the table displays the node name and snapshot memory type and time stamp. To the right of each header is a , which expands one side or the other to cover the entire viewport, and selecting it a second time (when it has become a) returns to the dual-table view. In the dual-table view, the data from each snapshot appears in a side-by-side comparison. A row with a white background indicates no change; blue means a row was added; green means a row changed; and red means a row was removed.

Configure View

The Configure View tab contains controls for the snapshot display in other tabs. The Custom select for full-memory snapshots area of the Configure View pane allows the user to select which snapshots in the project are available for selection in the Structure Map tab and for investigation and comparison. Choosing Default selects them all; Clear All selects none of them. The Default button in the bottom right restores all settings to the default values, and the Apply button next to it configures the view to match the preferences selected.

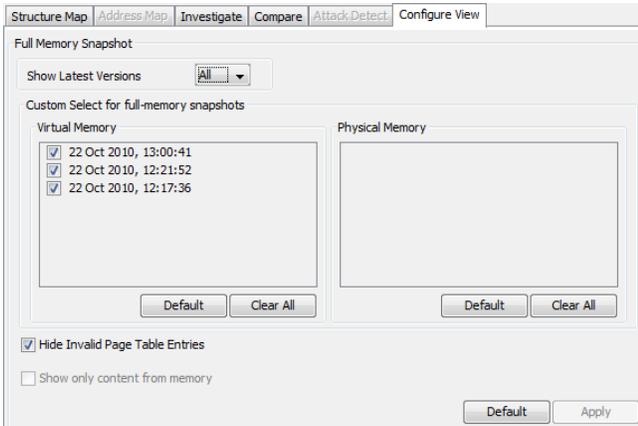


Figure 14: The Configure View tab.

Licensing and Configuration

The Options menu offers access to WindowsSCOPE configuration dialogs.

Selecting Options»License Manager... brings up the dialog shown in Figure 1, displaying license information. Selecting the View License Details button brings up an additional dialog displaying details of the license issuer and holder. If you have an expired trial license or wish to purchase a permanent license, you can follow step one to purchase or register for a license. After you have purchased or registered for a license you can follow step 2 to request your license file. The license file that you request will be sent to you in an email within an hour. After downloading the license file attachment from the email, you can use the Install License button to locate the license file and install it in WindowsSCOPE. Selecting Options»Set Profile... brings up the dialog shown in Figure 15 (left), allowing the user to set the Analyst and Company fields of all future snapshots.

Selecting Options»Configure Repository... brings up the dialog shown in Figure 15 (right). Clicking the Browse... button allows the user to select the project repository directory.

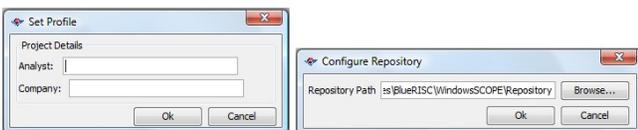


Figure 15: Profile dialog (left), Repository dialog (right).

Glossary

The following terms are defined in the context of this guide. In a definition, italic text indicates a reference to another term within the glossary.

ANSI — in WindowsSCOPE, a type of character that is eight bits wide, i.e., occupies one byte in memory. The acronym stands for American National Standards Institute, and on Microsoft Windows the ANSI encoding or ANSI code page (ACP) refers to the Windows-1252 character encoding, which contains the Roman alphabet, Arabic numerals, and standard English punctuation, among other things.

architecture — a set of instructions and low-level features and specifications for a processor. For example, x86 is an architecture designed by Intel with a large set of 32-bit (four-byte) instructions and a small set of 32-bit registers, among other specifications.

assembly — machine code represented in a human-readable set of instructions and operands.

bookmark — in WindowsSCOPE, an indication that a particular row in a table has been marked by the user. Bookmarked lines have attached comments.

code — binary-encoded instructions for the processor contained in a module, such as an executable or a DLL.

disassembly — the process of converting binary-encoded instructions into human-readable assembly code.

DLL — a shared library of code. The acronym stands for Dynamic-Linked Library.

driver — in Windows, a program designed to allow high-level software to interact with hardware.

exception — a special condition that changes the regular flow of program execution.

file handle — an object used by an application to access a file on a disk.

flow diagram — in WindowsSCOPE, a diagram constructed to indicate the flow of assembly code. A flow diagram has two levels: one level indicates the relationship between procedures (represented as ovals) using arrows; the other level indicates the relationship between blocks of code within each procedure (represented as rectangles containing assembly code) using arrows.

frame number — a label for a section of physical memory (a frame). Data from virtual memory which is currently loaded into physical memory is in a frame, and the page table uses the frame number to convert the virtual memory address to the physical memory address.

function — see procedure.

handle — see file handle.

IAT — a table of pointers to functions that have been imported from external DLL files. The acronym stands for Import Address Table.

IDT — a structure in memory used by Windows to support handling processor exceptions and interrupts. It contains pointers to each interrupt handler so that the processor can handle the interrupts independently. The acronym stands for Interrupt Descriptor Table.

interrupt — a signal indicating the need to stop program execution and begin executing code in an interrupt handler.

IP — a packet-based protocol used to transmit data. The acronym stands for Internet Protocol.

IP address — a set of four numbers, each within the range 0–255 (i.e., one byte), used as a label for a network device.

kernel — the central component of an operating system, responsible for bridging user applications and the processor and other hardware.

kernel-mode — describes a program that is granted system-changing privileges by the processor. In Windows, kernel-mode programs are drivers.

license file — a file containing the data necessary to authorize WindowsSCOPE use on a machine, i.e., containing the license.

memory forensics — investigation of a snapshot of a computer's memory in order to determine useful information about the computer's state at the time of the snapshot.

module — in WindowsSCOPE, a block of code in a program, including the program itself and code imported from other programs and libraries.

network socket — a mechanism provided by the operating system by which an application can communicate over a network. Each socket is tied to a single IP address, port, and process.

offset — in computer architecture, a memory offset is a number indicating a memory address relative to some known position in memory rather than an absolute address.

PAE — a feature in an architecture allowing more memory to be addressed than would normally be allowed due to the size of a memory address. For example, 32-bit addresses can only refer to a maximum of 4GB of memory, but a 32-bit system with PAE could address at least twice as much. Analogous to a system allowing mail delivery to more than 100,000 areas when ZIP codes are still five digits. The acronym stands for Physical Address Extension.

page — a fixed-length block of virtual memory. Virtual memory is loaded to and stored from physical memory in pages. On most systems, one page is 4,096 bytes.

page directory — a table used to translate virtual memory addresses into physical memory addresses. A page directory actually refers to the appropriate page table for the exact address.

page directory pointer — a pointer to a page directory. A PAE system has one such pointer for each process. A non-PAE system has only one page directory and thus has no page directory pointers.

page table — a table used to translate virtual addresses to physical addresses.

physical memory — actual random-access memory in a computer (physical microchips and integrated circuits).

port — in IP networking, a port is a number associated with a particular process, allowing multiple processes to communicate independently of one another using the same IP address.

procedure — a set of instructions, often given information to process and often giving information back to its caller. Procedures are a convenient means of reusing code in an application, so one task which must be executed many times may only be stored once but called each time it is needed.

process — an instance of a program being executed. As a program is only a set of instructions, it may be executed multiple times simultaneously; each execution of a program is a process.

project — in WindowsSCOPE, a set of snapshots.

registry — in Windows, the registry is a global database for storing settings for the operating system and applications.

registry key — in Windows, an entry in the registry.

repository — in WindowsSCOPE, the location on disk containing accessible projects. By changing the repository location, a different set of projects is made available.

snapshot — in WindowsSCOPE, a collection of information gathered from a computer's memory at a particular time. Ideally, a snapshot represents an instant; however, building a snapshot takes time, so the instant is not exact.

socket — see network socket.

SSDT — a table of pointers to functions exported by the kernel, used for completing system calls at the kernel level. The acronym stands for System Service Descriptor Table.

string — an ordered set of characters; continuous text.

symbol — in Windows, a named component of a program or library.

TCP — a protocol for transmitting data over IP. TCP implies a two-way connection. The acronym stands for Transmission Control Protocol.

text — see code.

UDP — a protocol for transmitting data over IP. UDP implies a one-way stream of information. The acronym stands for User Datagram Protocol.

Unicode — a universal consistent encoding standard for characters allowing for most of the world's writing systems to be represented electronically.

user-mode — describes a program restricted to a safe subset of instructions, not granted system-changing privileges.

virtual memory — a memory management technique allowing an operating system to give each application the illusion that it is operating on a contiguous section of memory and allowing the system to use more physical memory than it has by storing portions of the virtual memory on disk.