IRBAS www.irbas.academyirmbr.com December 2014
International Review of Basic and Applied Sciences Vol. 2 Issue.12

Gathering Data from Client Where the Monitoring Application Is Installed

SALMAN HUSSAM
Faculty of Computers and Automatics University of Polytechnics Romania
Email: phd@hussam.ro

Abstract

Scientific tracking of time is becoming a major need due to the diversity of social media websites / applications. That are seriously affecting the time people spend at work, so people are starting to spend more time at work with less work done. So "Logger" is an application that will help us do so. (Tracing the time spent at work), there is a very large data content that we can monitor and store from a client (monitored computer device) some of these data is not really important to gather as it doesn't help us in studying the behaviour of the client, so know what data to select and how to gather this selected data is something very important for the good functionality of the monitoring system, in this paper i will present the most important data to gather and how to gather it properly.

Key Words: *Client, Monitoring, Installed.*

Introduction

Einstein rocked the world of science with the idea that " everything is relative" One can hardly imagine the same thing having the same effect today, where relativity is a daily fact. Take time, for example. There are myriads of ways to spend your time, in any possible circumstance. People are listening to audio lessons while walking on the street, playing in the subway, buying things online while waiting in traffic and, of course, spending vast amounts of time on the Internet, at home and at work.

It seems obvious that the need to accurately measure time spent on different activities is paramount in order to efficiently manage one's life and business, yet there are very few tools that actually facilitate that. not only tools showing the time spent, but also analyzing the gathered data and combining information from as many sources and people as possible to allow for a scientific tracking of time.

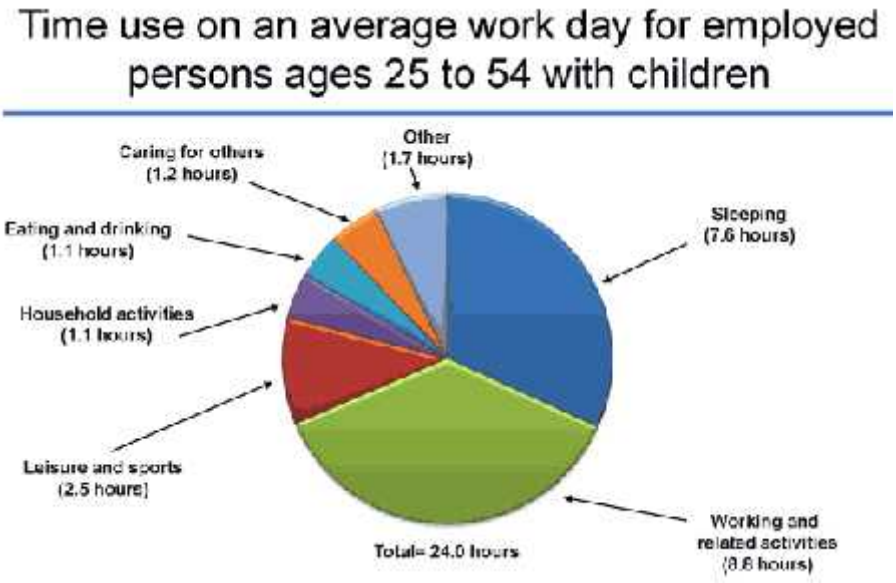
I will input here as data about statistics on how many hours people spend at work, how many are actually work, internet browsing habits, time used by people at home on computers, the lawsuits on employers not paying the time it takes for computers to boot up or for workers to prepare for work before actually performing it, ..etc

On average, men spent seven more hours per week (39.7) in the workforce than women (32.8) in 2007. Between 1976 and 2007, the number of weekly hours declined for men and increased slightly for women. On average, in 2007, men spent 60 minutes less at work, while women spent 18 minutes more at work than they did in before 2007

Average hours Worked per week, by number

The American Time Use Survey collects information about the activities people do during the day and how much time they spend doing them. For example, on an average day in 2011, Americans age 15 and over slept about 8.7 hours, spent 5.2 hours doing leisure and sports activities, worked for 3.6 hours, and spent

1.8 hours doing household activities. The remaining 4.7 hours were spent in a variety of other activities, including eating and drinking, attending school, and shopping.



These numbers are for all persons in the U.S. age 15 and over, and for all days of the week combined. The information can be further analyzed by age, sex, employment status, day of the week, or presence and age of household children. Looking at information for certain groups in the population provides a more accurate picture of how each group spends its time. For example, the chart above shows how employed persons ages 25 to 54, who live in households with children under 18, spent their time on an average workday. These individuals spent an average of 8.8 hours working or in work-related activities, 7.6 hours sleeping, 2.5 hours doing leisure and sports activities, and 1.2 hours caring for others, including children.

However, Lochhead said, it must be remembered that there is an element of individual choice in spending more time at work than with family, and the study, above all else, shows that Canadian workers are committed to their jobs.

Younger children mean more family time

It found that the estimated time spent with family by workers with a child younger than five is much greater than that of workers who had a spouse but no children. Those with young children spent about one hour more with their families.

"In fact, when children, especially young children, are present, women spend significantly more time with family than men do," the study says.

Single parent workers with a young child spent the most time with family, but single parent workers with older children spent the least time with family.

The study found that workers who worked long hours spent relatively little time with their families.

I will be speaking in this essay about " Data Gathering" :
Gathering the running processes data
Gathering the file data
Network connections

Computer use: mouse and keyboard
Desktop screenshots
Computer identification data

Gathering the running processes data

On a windows XP machine, there are several ways to gather the running processes data. One of them is to get the process id and its handle for any running process by using functions available in PSAPI.DLL. This Dynamic Link-Library (DLL) is distributed with the Microsoft Platform Software Development Kit (SDK). As example, one could use the performance objects of this SDK to get the current processes by reading the counter "ID Process". For this the function GerProcessID is called to retrieve the process id for the required process name.

Because in this project I have user C# technologies, there is a namespace that has several functions which provides the means to manage processes, thread, eventlogs and performance informations on a windows machine. This namespace is "System.Diagnostics". The "System.Diagnostics.Process" object will also provide access to certain ways of managing the machine's system running processes. Some usefull properties, methods and events of this class are:

- Properties:
  - StartTime (Shows the time that the associated process was started)
  - TotalProcessorTime (Shows the amount of CPU time the process has taken)
  - UserProcessorTime (Shows the user processor time for this process)
  - Threads (gives access to the collection of threads in the process)
  - ExitTime (Shows the time that the associated process exited)
  - Handle (Shows the native handle of the associated process)
  - HandleCount (Shows the number of handles opened by the process)
  - Id (Shows the unique identifier for the associated process)
  - MachineName (Shows the name of the computer the associated process is running on)
  - MainWindowTitle (Shows the caption of the main window of the process)
  - NonpagedSystemMemorySize64 (Shows the amount of nonpaged system memory allocated for the associated process)
  - PagedMemorySize64 (Shows the amount of paged memory allocated for the associated process)
  - ProcessName (Shows the name of the process)

- Methods:
  - GetProcesses() (This gives a list of process resource on the local computer)
  - GetProcessesByName(String) (This method returns an array of new Process components and associates them with all the process resources on the local computer that share the specified process name)
  - GetType (Provides the Type of the current instance. This method is inherited from class Object)
  - OnExited (This method will raise the Exited event, on which the program can log the exit time of the process)
  - ToString (This method formats the process's name as a string, combined with the parent component type, if applicable. An important note is that it overrides the Component.ToString() method of "System" namespace)

- Events
  - Exited (This event occurs when a process exits)

Besides the methods and properties of the Process class that allows monitoring local or remote a certain computer, it also provides control methods, on which I will not be focusing on because a logger application had the purpose of monitoring a machine and not interfere with its processes.

The “Process” class provides access to any process that is running on a windows machine. A process is a running application which is turn can be executed by a thread. A thread is the basic unit to which the operating system (windows) allocates processor time and can execute any small part of the process.

To use the “Process” class methods, first you need to initialize a new Process object, after which the methods and properties provide easy access to the information about the running process. This information includes the set of threads, the loaded modules (executable files “.exe” or libraries “.dll”), and performance information like the amount of system memory the monitored process is using.

A very important aspect of monitoring the system’s processes is that if the system is running both 32-bit and 64-bit processes you will need two monitoring applications, for a 32-bit process can’t access the data of a 64-bit process. This issue will throw a “Win32Exception” exception.

For Windows, a system process is uniquely identified on a certain system by its unique process identifier. Beside it’s identifier a process has also a handle. The difference is that the handle might not be unique on the system. A short explanation is that a handle is a resource identifier. The usefulness of the handle is that the operating system persist the process handle after the process has been stopped. By using this, one can get the process’s admin information such as the ExitCode and the ExitTime, which are extremely useful when monitoring processes. Because handle is a type of resource which is very important, leaking handles proves to be more virulent that leaking memory.

As detailed above, for running the process data gathering I used the Process class. It describes all information available about a running process and it lists the running process list by the use of the GetProcesses() method. The current process can be determined by the GetCurrentProcess() method.

The third part has proven so tricky that in the end it was almost abandoned. While the system is quite willing to give you information about the superficial aspects of a running application, it proves quite stubborn in allowing you access to the internal functionality of it. Also, in the luckiest cases even the most common of applications have proprietary and hard to use APIs, while most of them do not have any.

In the first version of “ Logger “ the only information I got was the Internet Explorer information on open sites and that only because Microsoft actually provides an API for it in the Interop.SHDocVw.dll and Microsoft.mshtml.dll libraries. Even so, in order to link the information from these APIs to the one gathered by the Process class I had to use the so called kernel functions, using the operating system directly.

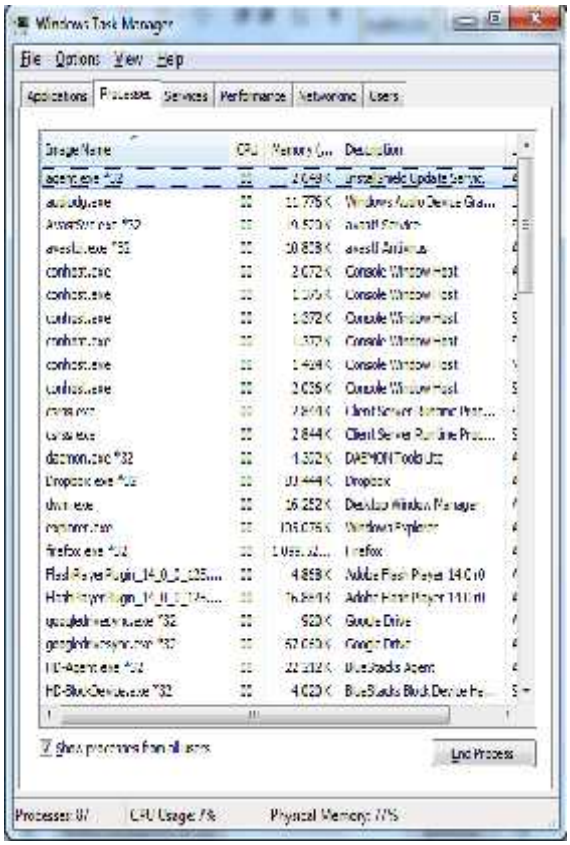
The information gathered for each process includes:

- the system process id, an integer value that is unique for a process in the list of running processes. Resetting the system will, of course, duplicate the id in the database.
- the window title, very important for a human in determining what the application actually is
- the process name, which is usually the name of the executable file
- the full path of the executable filename
- execution start time
- arguments used for the execution
- is it the active process or not

==========

The tests have allowed for gathering a lot of data and it showed things that were not taken into account at the time the application was designed.

The kernel method that I had used to list the open files did not only show files, but all the file handles that were opened, including devices, registry keys and special internal kernel BaseNamedObjects



Even for an idle computer, the number of such kernel files, temporary files or log files averages 700 , and can reach 4500 during intense usage.

Gathering the file data

The list of files created, changed and deleted was easy with the .Net class FileSystemWatcher. Its constructor receives the full path of the folder to monitor and then it exposes events on creating, renaming, changing, deleting and on file errors.

Since it must receive a folder path as a parameter the DriveInfo.GetDrives() method was used to get a list of all the drives and instantiate a FileSystemWatcher for each one.

Here the problems started. One cannot put a monitor on the floppy disk drives without the annoying head move sound and the little light in the front lighting up. So I had to remove the drives A: and B: from the list.

That doesn't solve the issue of possible USB floppy drives or other types of drives that don't cache the situation of the filesystem (like some older CD drives).
Even so, the issue of temporary drives must be addressed, like when someone uses a stick drive. I did that, but only on a periodic basis, without a plug in the mechanism that actually mounts or unmounts a new drive.

Even worse, while the FileSystemWatcher class says what the files are and what operation was performed on them, it says nothing about the process that performed the change. I still haven't found a solution for this and for version 2 of " Logger " I indend to put aside this simple to use, yet so blunt instrument.

Then there came the issue of open files. There is no .Net class to encapsulate the querying of opened files. There are some solutions of using system kernel functions to list the files, but then one can only access the information on the files opened by the same process as the one makeing the query. In order to get all the information (like in the free Process Explorer application which lists all processes and opened files, as well as their opening process) one must hack the system, loading a low level driver that caches the information for the files in order for it to be queried.

That brings me to the first use of another language, that is C++, because .NET is an intermediate language and cannot be used for kernel drivers in Windows XP. In order to encapsulate the functionality of a low level driver in my .NET application I used the same trick Process Explorer uses, that is embed the compiled driver as a file resource, then save it as a temporary file (I used Path.GetTempFileName() to get the temporary file name) and install it from there.

So I had the files as they were changed, but without knowing what changed them, and the files that were opened and by whom, without knowing when they were opened.

After all this work, the information gathered for each file consists of:

- type of change, an Enum which says the file was changed, opened, renamed, etc. plus two more status codes: new drive and removed drive
- full path of the file
- new path of renamed files
- exception raised when trying to open it
- process id (only for opened files)

FileItemType enumerator

The enumerator in the above figure contains the type of changes that can occur of a certain file. Thus, the application will record all the changes that occur on each file of the monitored system.

FileItem Class

The above figure presents the structure of the FileItem Class which has the following properties to access the private members:

- ChangeType – sets or returns the FileType of the monitored file
- Exception – returns the Serialization exception if it occurs
- FullPath – a string that contains the full path of the monitored file
- NewPath – a string that contains the new path of the monitored file if the file is changed
- ProcessId – an integer number that contains the process Id of the application that changed the file

Network connections

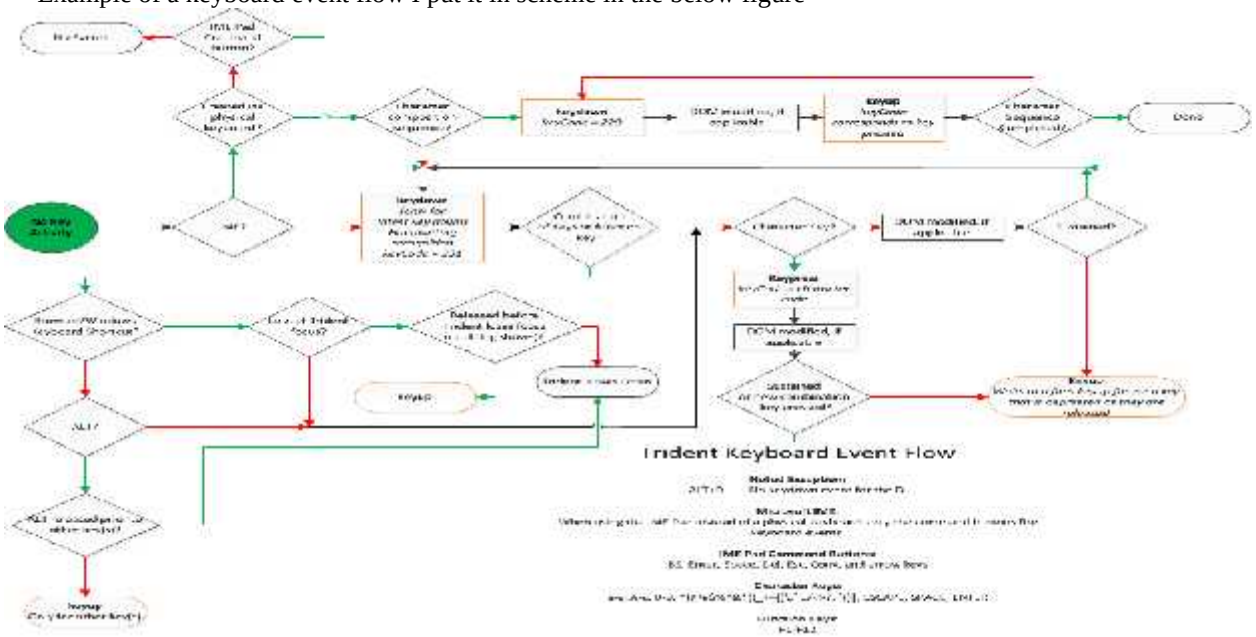
The NetworkAddressChanged and NetworkAvailabilityChanged events were used to switch within the program , configuration data of the network card and the flag that will allow data sending over the internet to the database server. Also IPGlobalProperties.GetIPGlobalProperties and then GetActiveTcpConnections() enumerated the open connections at any time. I also monitored the physical address (also known as a MAC address, it will be used in identifying computers) and The total bytes received or sent from the last adapter reset

And here in order to insure system security , and to insure that each client will be uniquely identified , I have created a unique (identifier) of the type string , which is the combination between MAC and USER and IP and I have identified the client by this number !

- Either he changes one of the 3 elements used , he will be still identified by the other 2
- If many computers are connected to the same IP router will have same ip! But diferent User and MAC

Computer use: mouse and keyboard

Example of a keyboard event flow I put it in scheme in the below figure



Almost all Windows programs and process are event based.

That's why, when any input device provides a signal, event is fired to catch. The main two input devices any computer has are a mouse and a keyboard. The mouse triggers only a hand full of events, like motion and motion speed, the two main buttons (left and right click), the scroll wheel which can also have from one to three buttons, and other buttons that can provide usefulness actions to a user. But comparing to the events a mouse can throw, a keyboard has much more keys. Besides the alpha-numeric keys, *it also has several buttons that can change the actions of the common keys when pressed*
some of The data on the keys and mouse movements consists of:

- **The rate at which the keys are tapped or the mouse moved can indicate a lot of things**
- **key value**
  - ***key up or down***
  - ***Key pressed or un-pressed***
- **mouse location**
- **mouse buttons pressed**
- **mouse click**

Desktop screenshots

It is said a picture is worth a thousand words !
Actually, for the simple task of employee monitoring, the active task and the screenshots are all that is needed.

Graphics.CopyFromScreen method. Was Used
The problem occurs in storing it in the database
package and making it as small as possible to send through network, so for this I have used and modulated another OpenSouce aplication for Resizing my secreenshots before sending .(OPEN CV)

Results

1- Testing have allowed me to find out that Even for an idle computer, the number of open kernel files, temporary files or log files averages 700 , and can reach 4500 during intense usage and is NOT only the activities that are shown in task manager .

2- Taking Screenshots with an interval of time less than 1 screen shot per minute is affecting negatively the CPU usage of the computer and also the internet consuming and speed of download / upload while client sends these screenshots to the server.

3- The unique (identifier) of the type string , which is the combination between MAC and USER and IP that has been created in in order to insure system security , and to insure that each client will be uniquely identified , have shown effect while testing to change the IP of the client computer r the MAC address of his network card , we still could identify the client through the other 2 un changed elements

4- logging all mouse and keyboard activity is not necessary for obtaining an accurate knowledge about the Human behavior on the computer at that time specially **/key value**/ it is consuming the resources of the computer and increasing the size of the package that the client sends to the server , its enough to monitor : *key up or down* / *Key pressed or un-pressed* / mouse location / mouse buttons pressed / mouse click

Conclusion

1- the scientific trace of time for employees at work , is crucial for both the employee and the employer , it will give positive results for the company progresses and for the employee social life and it will add free time to human's life.

2- The use of the computer control devices not only determines things like what was done on the computer, but also if the computer is idle or not. Also, the rate at which the keys are tapped or the mouse moved can indicate a lot of things, from the state of mind of the user, to a possible medical emergency (in future plans).

3- Information about the position and speed of the mouse, while interesting from an academic standpoint, did not reveal a lot of information. What did fill up a great deal of the data package was the information on open processes and files. Mostly files, actually.

4- using the unique identifier is very important to have a strong control upon the monitored device , Either if the user(client) changes one of the 3 elements used to identify him(user,IP,MAC) , he will be still identified by the other 2 , So If many computers are connected to the same IP router they will have same IP! But different User and MAC.

5- even though its said that a picture worth thousands of words , getting screenshots in a frequently base affects the system negatively as we discussed. (using CPU , traffic , processing power to optimize it)

References

www.w3schools.com
http://en.wikipedia.org/wiki/.NET_Framework
http://msdn.microsoft.com/en-us/library/czefa0ke(VS.71).aspx
http://www.dotnetpowered.com/languages.aspx
http://msdn.microsoft.com/en-us/library/windows/desktop/bb530746(v=vs.85).aspx
http://raspberrywebserver.com/cgiscripting/sending-data-to-an-HTTP-server-get-and-post-methods.html
http://blog.oauth.io/oauth-tutorial
http://books.google.ro/books?id=VGT1_UJzjM0C&pg=PA9&lpg=PA9&dq=what+is+.net+programing+language&source=bl&ots=zqLHRILJMf&sig=ZCWqoPLTFcIS-OtC-w1bctQlYds&hl=ro&ei=aC8sSo7pCdLL_AapvLnwCg&sa=X&oi=book_result&ct=result&resnum=5