

BACHELOROPPGAVE:

**Behaviour Logging Tool - BeLT**

FORFATTERE:

Robin Stenvi

Magnus Øverbø

Lasse Johansen

DATO:

15.05.2013

## Sammendrag av Bacheloroppgaven

Tittel:	Verktøy for logging av brukerinteraksjon		Nr: -
			Dato: 15.05.2013
Deltakere:	Robin Stenvi Magnus Øverbø Lasse Johansen		
Veiledere:	Dr. Hanno Langweg, Associate Professor		
Oppdragsgiver:	NISlab, Høgskolen i Gjøvik		
Kontaktperson:	Soumik Mondal, soumik.mondal@hig.no		
Stikkord	Biometri, Interaksjonslogging, Tastelogging, Personvern, Windows-programmering		
Antall sider:	Antall vedlegg: 8	Tilgjengelighet: Åpen	
395			
<p>Kort beskrivelse av bacheloroppgaven:</p> <p>BeLT er en applikasjon som logger hvordan en person bruker tastatur, mus og det grafiske grensesnittet på en PC, i tillegg lagrer BeLT informasjon om hardware og enheter tilkoblet pcen. Hensikten med programmet er å forenkle innsamling av data fra flere brukere – når brukerdata er innhentet sender programmet disse sikkert til en server for lagring. Dette er for å støtte NISlab sitt pågående forskningsprosjekt for å identifisere unike brukermønstre i hvordan enkeltindivider bruker en datamaskin.</p> <p>Vårt bidrag er utviklingen av BeLT og dens tilhørende klient-server arkitektur som muliggjør å samle og analysere data i større skala. Vi har programmert en transmisjonskomponent for BeLT som benytter seg av RFC 5424(Syslog-protokollen) og TLS(Transport Layer Security). Klient-server arkitekturen er skalerbar og serveren er optimalisert for å ta imot og prosessere data fra BeLT. For å kunne passe inn i dagens og fremtidens situasjonsbilde har vi gjort det mulig å lagre dataene på serveren i CSV, XML og relasjonsdatabaser. Med sikkerhet i tankene har vi implementert sertifikater i løsningen vår, ved å kodesignere applikasjonspakken og ved å ha et serversertifikat som blir brukt ved transmisjon.</p> <p>BeLT er forskjellig fra tidligere arbeider fordi det fanger opp interaksjon og bestemmer når det skjedde i forhold til tidligere interaksjon – dette muliggjør å se på hendelser som årsak-virkning. Virkninger er ofte endringer i skjermbildet, for å fange opp disse endingene har vi brukt Microsofts UIA (User Interface Automation). Ved å korrelere hendelser med hverandre åpner BeLT for nye måter å tolke data på – dynamikken i hvordan mus og tastatur blir brukt kan kanskje bli belyst på en ny måte når man også ser det i sammenheng med software?</p>			

## Summary of Graduate Project

Title:	Behaviour Logging Tool - BeLT		Nr: -
			Date: 15.05.2013
Participants:	Robin Stenvi		
	Magnus Øverbø		
	Lasse Johansen		
Supervisor:	Dr. Hanno Langweg, Associate Professor		
Employer:	NISlab, Høgskolen i Gjøvik		
Contact person:	Soumik Mondal, soumik.mondal@hig.no		
Keywords	Biometrics, Behavior logging, Keylogging, Privacy, Windows programming		
Pages: 395	Appendixes: 8	Availability: Open	
<p>Short description of the main project:</p> <p>BeLT is an application that captures mouse, keyboard and GUI (Graphical User Interface) interaction on a computer, it also provides information about the system state and hardware peripherals. The purpose of BeLT is to simplify data acquisition – after capturing data on a client BeLT sends it securely to a central server for storage. The data is planned to be analysed to develop a new way of finding distinct signatures in a users interaction with a computer. This development is currently a part of NISlabs research in biometrics.</p> <p>Our contribution is the development of BeLT and a client-server architecture that makes it possible to gather and analyse data sets in a larger scale. We have programmed a transmission component for BeLT that communicates with a server based on RFC5424 (Syslog protocol) and TLS (Transport Layer Security). The client-server architecture is scalable and we have optimized the server to handle and efficiently store the data received from BeLT. In order to meet current and future needs we have made it possible to store the data in CSV, XML and relational databases. For security purposes we have implemented certificates to ensure that both the application and the server communication is secure, by codesigning BeLT and by verifying the server identity before sending data from BeLT.</p> <p>BeLT is different from previous research projects because it captures interaction and correlates it with previous actions – this makes it possible to look at the data in a cause-effect perspective. Many of the changes on a computer is visible on the display, we have managed to capture this by using Microsofts UIA (User Interface Automation). By putting BeLTs captures in relation to one another, we open up for new research possibilities – analysing keystroke and mouse dynamics correlated with GUI interaction can possibly uncover currently unknown user patterns.</p>			



## Preface

BeLT has been developed by three bachelor students from Gjøvik University College in the spring of 2013. This project has been done to support NISlabs research on biometric signatures based on human behavior with a computer. We have put an effort into programming a secure, stable, efficient and correct program. We have also focused on making a secure client-server architecture.

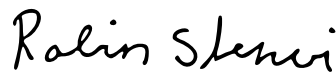
First and foremost we want to thank Soumik Mondal and his supervisor, Professor Patrick Bours, for their support and open-mindedness, they listened to our suggestions and always gave us essential feedback. We want to thank our supervisor, Associate Professor dr. Hanno Langweg for pushing our goals and for providing invaluable advice on how to approach the project.

Other people we want to thank are:

- Dr. Erik Hjelmås for loaning us the virtual server that we used for testing and implementing the server-side of BeLT.
- Professor Rune Hjelsvold for priceless information on how we best could design the database to handle a continuous and possible large stream of data.
- IT-department at GUC for giving us advice on how to apply for digital certificates.



Lasse Tjensvold Johansen



Robin Stenvi



Magnus Øverbø



# Contents

<b>Preface</b> . . . . .	<b>v</b>
<b>Contents</b> . . . . .	<b>vii</b>
<b>List of Figures</b> . . . . .	<b>ix</b>
<b>Code snippets and scripts</b> . . . . .	<b>ix</b>
<b>List of Tables</b> . . . . .	<b>x</b>
<b>List of Abbreviations</b> . . . . .	<b>x</b>
<b>Glossary</b> . . . . .	<b>xi</b>
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Background . . . . .	1
1.2 Project objective . . . . .	3
<b>2 Requirement Specification</b> . . . . .	<b>5</b>
2.1 Functional requirements . . . . .	5
2.2 Operational requirements . . . . .	5
2.3 Graphical design requirements . . . . .	6
2.4 External requirements . . . . .	7
<b>3 Theory and technology</b> . . . . .	<b>9</b>
3.1 Application . . . . .	9
3.2 Development . . . . .	14
<b>4 Design</b> . . . . .	<b>17</b>
4.1 Architectural design . . . . .	17
4.2 Implementation view . . . . .	17
4.3 Logical view . . . . .	19
4.4 GUI design . . . . .	24
<b>5 Implementation</b> . . . . .	<b>31</b>
5.1 Application . . . . .	31
5.2 Server . . . . .	35
5.3 Development . . . . .	41
5.4 Algorithms . . . . .	44
<b>6 Testing and analysis</b> . . . . .	<b>51</b>
6.1 Tests on client . . . . .	51
6.2 Performance optimization on server . . . . .	55
6.3 Server testing . . . . .	57
<b>7 Privacy</b> . . . . .	<b>71</b>
7.1 Anonymity of the user . . . . .	71
7.2 Confidentiality . . . . .	72
7.3 User awareness . . . . .	72
7.4 Abuse by authorized personnel . . . . .	73
7.5 Abuse by un-authorized personnel . . . . .	74
7.6 Transparency of logged data . . . . .	75
7.7 Storage of data . . . . .	75

<b>8 Conclusion</b>	<b>77</b>
8.1 Achievements	77
8.2 Requirement specification and results	77
8.3 Future Development	80
8.4 Alternatives	81
8.5 Evaluation of group work	81
<b>Bibliography</b>	<b>83</b>
<b>A BeLT: System Manual</b>	<b>87</b>
<b>B BeLT: User Guide</b>	<b>141</b>
<b>C Windows application certification</b>	<b>149</b>
<b>D Scripts</b>	<b>153</b>
D.1 Python script to calculate time statistics	153
D.2 Python script to measure mouse compression on file	153
D.3 Python script to paint mouse movements from file	155
D.4 SQL procedure for inserting data into database	156
D.5 Script to insert data into indexed database	158
D.6 Bash script used to run the server test	159
D.7 RAW part of Syslog-NG configuration file	160
D.8 CSV part of Syslog-NG configuration file	160
D.9 XML part of syslog-NG configuration file	161
D.10 Syslog-NG for database storage	162
D.11 Bash script for inserting data into database	162
<b>E BeLT: Source Code Documentation</b>	<b>163</b>
<b>F BeLT: EULA example</b>	<b>371</b>
<b>G Work Log</b>	<b>373</b>
G.1 Work activity documentation	373
G.2 Progress log	374
G.3 Meetings	376
<b>H Preliminary project</b>	<b>380</b>



## List of Figures

1	Implementation scheme of system architecture . . . . .	18
2	Logical view of the client application . . . . .	20
3	Logical view of the server application . . . . .	22
4	Initial system tray design . . . . .	25
5	Finished system tray design . . . . .	25
6	Initial application GUI design . . . . .	26
7	Final window view design . . . . .	26
8	BeLT GUI: Settings dialog(Basic/All settings) . . . . .	27
9	BeLT GUI: Display settings dialog . . . . .	28
10	BeLT GUI: Send local file dialog . . . . .	28
11	BeLT GUI: About BeLT dialog . . . . .	29
12	Mouse compression with 30 % of original dataset . . . . .	45
13	Mouse compression with 19 % of original dataset . . . . .	45
14	Mouse compression with 14 % of original dataset . . . . .	46
15	Mouse compression with 11 % of original dataset . . . . .	46
16	Depiction of our server test setup . . . . .	59
17	Flow chart for how the testing was performed . . . . .	60
18	Percentage of time spent idle when using raw mode . . . . .	67
19	Percentage of time spent idle when using CSV . . . . .	67
20	Percentage of time spent idle when using XML . . . . .	68
21	Percentage of time spent idle when using database . . . . .	69
22	ER-model of our database system . . . . .	69
23	Percentage of time spent idle when using indexed database . . . . .	70
1	Graph of summarized work effort . . . . .	373

## Code snippets and scripts

5.1	Mouse compression algorithm . . . . .	45
6.1	C++/pseudocode for time granularity testing . . . . .	53
6.2	Section 1 of logging program for time granularity testing . . . . .	53
6.3	Section 2 of logging program for time granularity testing . . . . .	53
6.4	C++/pseudocode for generating key events on second test . . . . .	54
6.5	Relevant part of syslog-ng.conf file . . . . .	56
6.6	Select all session without a "stop" event . . . . .	64
6.7	Select all distinct user ids from sessions without a stop event . . . . .	64
6.8	Check if anyone are missing multiple sessions . . . . .	64
6.9	Check if the session without "stop" event is the last session . . . . .	64
D.1	Python program to calculate time statistics on input file . . . . .	153

D.2	Python script to test mouse compression on files . . . . .	153
D.3	Python script to paint mouse movements on a graph . . . . .	155
D.4	SQL procedure for inserting data . . . . .	156
D.5	Indexed database script . . . . .	158
D.6	Script at the server used for testing performance . . . . .	159
D.7	raw part of the syslog-ng.conf file . . . . .	160
D.8	CSV part of the syslog-ng.conf file . . . . .	161
D.9	XML part of the syslog-ng.conf file . . . . .	161
D.10	Syslog-NG config for databse storage . . . . .	162
D.11	Bash script for inserting data to DB . . . . .	162

## List of Tables

1	List of events retrieved from Windows hooking . . . . .	10
2	Status indicators for system tray . . . . .	25
3	List of application tested with UIA, and our experiences . . . . .	32
4	List of events gathered with UIA . . . . .	33
5	CSV format for BeLT system-messages . . . . .	37
6	CSV format for mouse events . . . . .	37
7	CSV format for software events . . . . .	38
8	CSV format for key events . . . . .	40
9	CSV format for hardware messages . . . . .	40
10	Table of events and their corresponding relationship . . . . .	48
11	Summary of our performance test . . . . .	51
12	Output from performance test . . . . .	57
13	Summary of our test results. . . . .	65
14	Number of losses within each hour long test . . . . .	66
15	Requirements completed or not fully completed . . . . .	79
1	Summary of Microsoft's app certification reqs. . . . .	151

## List of Abbreviations

ACE	Access Control Event	CPU	Central processing unit
ACL	Access Control List	CSV	Comma Separated Values
API	Application Programming Inter- face	EULA	End User Licence Agreement
BSD	Berkeley Software Distribution	FIFO	First In First Out
CI	Continuous Integration	FQDN	Fully Qualified Domain Name
CIA	Confidentiality, Integrity and Availability	GUC	Gjøvik University College
		GUI	Graphical User Interface

HTTP	Hypertext Transfer Protocol	RAM	Random-access memory
HTTPS	Hypertext Transfer Protocol Secure	RFC	Request For Comments
HW	Hardware	SAR	System Activity Report
IDS	Intrusion Detection System	SDK	Software Development Kit
IP	Internet Protocol	SHA	Secure Hash Algorithm
LTS	Long Term Support	SQL	Structured Query Language
MD5	Message Digest 5	SSH	Secure Shell
MITM	Man In The Middle	TCP	Transmission Control Protocol
MSAA	Microsoft Active Accessibility	TLS	Transport Layer Security
NISlab	Norwegian Information Security laboratory	UAC	User Account Control
OS	Operating System	UDP	User Datagram Protocol
OSE	Open Source Edition	UIA	User Interface Automation
		WiX	Windows Installer XML toolset
		XML	Extended Markup Language

## Glossary

**childs** All elements one level below the current element in the tree 9

**descendants** Childs and grandchilds of the current element in the tree 9

**hierarchical tree** A data structure used to represent information by linking together several nodes 9

**infinitesimal** Number so small that we can't measure it 71

**parent** The element above the current element in a tree 9

**siblings** All element which is on the same level as the current element. 31

**terminal based applications** All applications that uses a text based interface, where you can write commands 33



# 1 Introduction

Our assignment was to deliver a tool to NISlab (Norwegian Information Security laboratory) , a whole package that could help them capture user data for use in biometric experiments. We have used an incremental system development model with iterations of two weeks. In our pre-project we defined goals for each week, with a total slack of one month at the end. We used the slack time solving unforeseen challenges and to improve the thesis report.

The audience of this report are those who are going to use or develop BeLT in the future, because this report serves as a complete documentation of what we have done and the theory behind it. Some interesting parts of the report is how to capture behavioral information in Windows, how to build a Windows Installer with WiX and how to securely implement a logging service.

## 1.1 Background

In this section we will discuss the background for this project and previous work related to it.

### 1.1.1 Project background

Soumik Mondal is working with an algorithm that can recognize unique patterns in how a user interacts with a computer. This algorithm is part of a research project where the goal is to continuously authenticate users based on their behavior and interaction with a computer. This work is part of Soumik Mondal's PhD project for NISlab.

To create this algorithm it is necessary to analyse a large amount of information about multiple users regarding how they interact with their computers. This data includes key-presses, mouse interaction and software interaction. When a user performs a task on a computer there are multiple ways to do the same task – for example the user can use keyboard shortcuts instead of clicking on objects on the computer or vice versa. We need an application that can see the context in how the user acts. When the user does a mouse click for example, the program should capture what was clicked, where it belonged to and the effect of this action.

It is important that the information we gather reflects the natural behavior of the users because we want to differentiate them. If the gathering of data was done in a controlled environment with only one type of computers, the test users wouldn't use the computers completely with their own preferences. There was a decision to make an application that could log behaviour directly from a personal computer, in the users natural habitat. This is the main reason why the BeLT is built for Windows, because most users use it. By doing this we are certain to obtain the most natural type of behavior.

Because the program runs on personal computers, there is a need for a transmission service to gather the data and this has been an important aspect of our project. Finally, we should ensure unobtrusiveness – BeLT should be easy to install and should not bother then user when it's running.

### 1.1.2 Previous work

There has been done substantial amount of previous work to capture mouse events[1, 2, 3], in most cases this is used to identify user activity with the goal of enhancing the user experience. The previous papers are written with web applications in mind and focus on visualizing how someone interact with a web application.

Some interesting points in [2], is the way to analyse mouse events, and the theory of how to infer abstract information from concrete information. They have divided it into four groups:

**Small and concrete:** Information you can infer directly from looking at the data.

**Small and abstract:** Information you can deduce by looking at the logs. Like something you think the user did because of a series of events.

**Large and concrete:** Certain information about a group of events.

**Large and abstract:** General information about the user.

RUI[4] is a program written in C# for Windows and Carbon Framework for Mac OS X. The program only store keyboard interaction and mouse interaction. It does not store any interaction with software. One noteworthy piece of information is how much data they saw on mouse movements. With heavy use of the mouse, it could be around 22 KB/m, but with continuous movements it could be as much as 156 KB/m.

AppMonitor[5] is an application that stores user interaction to find out how the user interacts with the computer. It uses Windows hooking to capture mouse and keyboard interaction and it uses Microsoft Active Accessibility to capture how the user interacts with software. AppMonitor only supports Adobe Reader 7 and Word 2003. This work is very similar to what we are trying to achieve and it has been a useful resource. The software was available by contacting the authors, but we didn't take advantage of this opportunity, since we had a different approach to the problem and the article gave us a nice overview of their methodology and experiences. Some experiences they had, which we think is important to consider is; if the user could, in real-time, see what was logged, they felt that the program raised few or no privacy concerns. They also reduced privacy concerns by not logging any regular typing<sup>1</sup>.

USim[6] is an IDS (Intrusion Detection System), based on detecting anomalies in human behaviour when interacting with a computer. This tool is specifically targeted with graphical user interfaces in mind. It specifies several interesting aspects of how to analyse the data, and what type of information you can deduce from the data.

WIDAM[7] is a monitoring tool implemented with Java and JavaScript, intended to run in the browser. This tool is similar to our tool, in terms of goal, but it runs in the browser which is quite different from what we are trying to achieve.

Something that is missing from all these papers are a generic way to capture behaviour information in Windows. Most only store mouse and keyboard interaction, AppMonitor, which stores software interaction, only supports two application. This report and our application will provide a way to gather information, both from input devices and from all software.

---

<sup>1</sup>Keyboard typing that can be used to reconstruct parts of or the whole text

### 1.1.3 Our background

We study at the bachelor in information security programme at Gjøvik University College. In this program we learn programming, operating systems, database theory and information security. The programme is similar to informatics and computer science bachelors, but it is more focused towards information security.

In our thesis project we applied almost every aspect of what we have learned – we have programmed an application, designed a database, worked with different operating systems and also applied security specific theory. The users privacy, anonymity and integrity was important for us when we programmed BeLT.

## 1.2 Project objective

### Effect objective

It is expected that the finished program should accomplish the following:

- Ease the process of gathering behavioral information.
- Collect more behavioral information, both from more users and in a larger timeframe.
- Collect more accurate information and place it in the correct context so that analysis will be easier and more effective.

### Result objective

The application shall be delivered by 15. May 2013 and should have the following functionality:

- Collect keyboard interaction.
- Collect mouse interaction.
- Collect software interaction.
- Send and store the information on a central server.
- Export the information to a sensible format

Our task is to create an application that collects information about how the user interacts with a computer. This information should be sent and stored at a central server. Where it can be retrieved in an easy and understandable format.

Another part of our task is to investigate different methods of gathering information about how the user interacts with applications. The method used should be generic, in the sense that we should be able to gather information from unknown programs. This will also guarantee that we don't need to update the program when a new version of other applications are released. If it's not feasible to make a generic method, the application should at least capture information from the following programs:

- Microsoft Word, Excel and Outlook.
- One or more PDF-readers
- One or more web-browsers
- Skype

The application should be designed a way that this list can easily be extended later. The task can be divided into six parts:

**Collect keyboard interaction:** Here we first need to investigate what is the best way to capture this information. It is important that the times are stored with millisecond accuracy. The method used must be effective and compatible on most Windows platforms.

**Collect mouse interaction:** Here we have the same tasks as above, but we also have one additional challenge. Mouse movements generate large amounts of data. RUI[4] which was mentioned earlier stores about 22 KB/min just in mouse movements, this data must be stored and sent across the network, a decrease in this value would be beneficial.

**Collect interaction with software:** We need to find a method to detect the changes that can happen in our software. Especially user actions that can be executed in different ways should be distinguishable by the logs. This method should be generic and work on all programs.

**Store information:** We first have to develop a format for storing the data. Then we need a solution to transmit and store this data on a central location. This solution must preserve confidentiality, integrity and anonymity in all phases.

**Export information:** We have to develop a useful format to present the data, this can for example be CSV or XML. The format should be human-readable, not in binary-format.

**Risk analysis for use of application:** We need to evaluate the possible risk by distributing this to multiple users, where everyone connects to the same server. We should also present mitigating measure to handle this risk.

Another important area is the time granularity. When the user does something, the timestamp of that event need to be as accurate as possible, down to milliseconds accuracy. Clocks on computers are not made to be completely accurate and if you gather a value that is measured in milliseconds, it doesn't necessarily mean that it is updated every millisecond. We also have to take into account differences between systems.

The project should also look into information security – threats and risks to the information gathered and user confidentiality and anonymity. Legal obligations set by the Norwegian law should be identified and met.

### **Out of the project scope**

- Analyse of the data captured
- Touchscreen support



## 2 Requirement Specification

### 2.1 Functional requirements

#### 2.1.1 Data capturing

BeLT must capture when keys are pressed and released. It should also capture system and specific program hot-keys. In the windows operating system there are some system level hotkeys that invokes special system behaviour, such as the windows key or Alt+F4. BeLT must be able to capture these with the exception of the Windows secure attention sequence.

BeLT must capture at minimum events in form of movement, button press, button release and interaction with the scroll wheel from the users mouse and touchpad.

Since Windows 8 is built for use with touch interfaces and touch gestures, the program must be programmed so that any future development to capture this interaction is possible.

The application must be able to distinguish between the different hardware(HW) components(HIDs, devices and screens) connected to the computer. It also must be able to capture when a change occurs on/with the HW.

Relative to mouse/keyboard capture the program should also capture what programs that are used, and what buttons/icons that is pressed inside the program. In addition the software should be able to log the CPU/RAM usage and any inserted/removed external peripheral equipment.

The system must at a minimum be able to store the captured data in a CSV file format depicted by the employer to meet his need for a file format.

BeLT must be able to correlate new events with past events and relate them to each other based on where in the application it occurred and what type of event has occurred. The relations we must identify is described in section 5.4.2 and in table 10.

BeLT has to capture information about as many applications as possible using a general capture mechanism, but BeLT must be at the minimum able to capture information from the following applications without problems.

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Adobe PDF Reader
- Microsoft Excel
- Microsoft Word
- Microsoft Outlook
- Skype

### 2.2 Operational requirements

#### 2.2.1 Stability, accuracy and security

- To measure biometrics the accuracy of the logging is crucial. The program should log events with millisecond accuracy.
- The system must have the capability to detect lost packets in retrospect.
- Transmissions and communication between the client and server must be secured. This is to ensure the integrity and confidentiality of the data is maintained during transmission.

- BeLT is required to run on Windows 7 and later Windows OS.
- BeLT software should be signed with a code signing certificate issued by a valid certificate authority.
- There should be implemented mitigative measures to hinder a third party to pose as another user by providing false input.

### 2.2.2 Runtime

- The program should run unobtrusive. By this we mean that the program should not ask for user input when BeLT is not actively being interacted with. When the program is not in active use it should reside in the system tray.
- BeLT should not crash/freeze at all during runtime in a way which may transmit or project sensitive information.
- During transmission the loss of logged events can't be greater than 1% of the total amount of captured events.
- The system must handle several users, but it will be limited by physical networking capabilities and server capabilities. There should be an estimation of the total amount of users that the system are able to handle at the same time.
- The users must be anonymous and their ID must be persistent across sessions

## 2.3 Graphical design requirements

- Upon start BeLT should be hidden from the user and only be displayed through the icon in the system tray. BeLTs GUI interface should by default not display any events.
- The user should be able to filter what data to show in the display field within the GUI through BeLT.
- The user should be able to hide BeLT by clicking the minimize button and through the menu bar.
- The logging controls for BeLT that should be implemented are "start", "stop", "pause" and "resume".
- BeLT must have a menu bar in the main window that holds items with the same functions as the buttons in BeLT, in addition an "exit" item, an "About BeLT" item and an item for the filter settings dialog.
- There should be a display field that shows the captured events in real time, based on filters settings set by the user.
- BeLT must incorporate an icon in the system tray area. This icon must display the current state of BeLT, are able to restore the window and control BeLT.
- The GUI should display some statistics about the users current session. It should display statistics about the total number of mouse clicks and keyboard presses.

## 2.4 External requirements

### 2.4.1 Legal requirements

The Norwegian law on privacy(Personopplysningsloven)[8] sets restrictions on many aspects when it comes to information gathering and processing. Even, though many of these don't directly apply to our project it is necessary for us to be aware and work within the confinements of the law, which can be represented by the following legal paragraphs.

- According to §33 is no consensus from Datatilsynet (The Norwegian Data Protection Authority) required to be granted if the data is voluntarily submitted to the collectors. This is achieved by users accepting an EULA(End User Licence Agreement).
- §8 describes the "Terms for processing personal information". We are allowed to gather the information based on it being voluntarily provided by the registered.  
*"Personopplysninger kan bare behandles dersom den registrerte har samtykket, eller..."*  
*"Personal information can only be processed if the registered has voluntarily agreed to it, or..."*
- §9 states one can gather this information if it complies with §8 in addition to one of the elements in §9. We fall under §9 item a:  
*"a) den registrerte samtykker i behandlingen"*  
*"a) the registered accepts the processing of sensitive personal information"*
- §13, "Information security", states that one has to assure that there's been taken adequate mitigative measures to handle information security in terms of CIA(Confidentiality Integrity and Availability). This has to be documented and provided to the respected authority, the Norwegian Data Protection Authority[9](*Datatilsynet*).

### 2.4.2 Ethical Requirements

#### General ethical guidelines

- All attendants must be informed of BeLTs purpose and terms of use. They must also accept BeLTs EULA before being allowed to use the application
- BeLT should be transparent when it comes to what it is collecting of information and never hide what it is doing.



## 3 Theory and technology

### 3.1 Application

#### 3.1.1 Software Interaction

UIA (User Interface Automation) is Microsoft's API (Application Programming Interface) to retrieve and send information to application. It is often used to automate tasks involving the GUI (Graphical User Interface). It is also used to provide accessible interfaces to people with disabilities. It is supported on Windows XP SP3 and newer[10].

UIA[11] is the technology that has replaced MSAA (Microsoft Active Accessibility). UIA provides some backward compatibility with MSAA and tries to eliminate the limitation in MSAA. The design and goal of both technologies are the same, provide information about the interface to the user and provide an opportunity to interact with the provider application<sup>1</sup>.

The UI (User Interface) elements are organized as a hierarchical tree, called the UIA tree. Childs and parent of the tree are based on processes, the root element is the desktop, its childs are usually applications, their descendants are UI elements in the application, like a textbox, button and so on. Here we can gather names, contents and several other useful bits of information. Another important part of our tool is the ability to subscribe for events, here we have the ability to be notified whenever a button is pressed for example[12].

When registering for events, there are four different categories[13]:

**Focus change** Event is raised whenever the user changes which element<sup>2</sup> is in focus.

**Property change** This event is raised when the property of an element in the tree, or the tree itself changes. Some examples are changing the text on a button or elements in the tree are rearranged. This happens indirectly because of something the user did, but on small user action can generate a lot of property changes, this makes it hard to correlate with the user action.

**Structure change** This event is raised whenever a change happened in the tree structure. This can be new button, new menu, close menu, anything that changes the UI.

**Other Events** This is a large category and can mean many different events, some important events are, button press, menu interaction, changes in edit fields and notification when new programs are started.

#### Notification without change

There are some events that may be raised even if the interface has not changed[14], see the list below:

---

<sup>1</sup>Applications that expose their interface are called providers, while application that retrieve and send information to the provider are called clients

<sup>2</sup>An element can be anything the user can see.

- `UIA_AutomationPropertyChangedEventId`
- `UIA_SelectionItem_ElementSelectedEventId`
- `UIA_Selection_InvalidatedEventId`
- `UIA_Text_TextChangedEventId`

This is not a big problem for us, but it is important to keep in mind that, even if we log an event, it isn't necessarily because of something the user did. We one register for the last event, so the remaining should not cause any problems.

### 3.1.2 Keyboard and mouse interaction

From Microsoft's documentation[15]:

A hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure.

There are many types of messages we can intercept, but we are especially interested in key presses and mouse actions.

Both mouse events and key events are divided into two categories, high level and low level. High level is useful to understand how Windows interpret the message. For example in high level we will receive a notification if there is a double click, while in low level we only get two fast clicks, but don't know whether it is a double click or not.

Information gathered with different Windows hooks				
Information	High level key	Low level key	Low level mouses	High level mouse
Timestamp	N/A	X	X	N/A
Press up / down	X	X	X	X
Scancode	X	X	N/R	N/R
Virtual key code	X	X	N/R	N/R
Context code	X	X	N/R	N/R
Transition state	X	N/A	N/R	N/R
Extended key	X	N/A	N/R	N/R
Injected	X	X	X	X
Number of key presses	X	N/A	N/R	N/R
Previous key state	X	N/A	N/R	N/R
x, y coordinates	N/R	N/R	X	X
Wheel	N/R	N/R	X	X
Handle to window	N/A	N/A	N/A	X

N/A - Not available, N/R - Not relevant, X - included

Table 1: List of events retrieved from Windows hooking

Table 1[16, 17, 18, 19] shows what kind of information we can get from various hooks, divided into low level key and mouse hooks and high level key and mouse hooks. X means available, N/A means not available, and N/R, means not relevant. Since mouse and key events generate different data, not all information is relevant to both. This table is meant as a reference for what interesting information we can get using different methods, it's not complete.

We only get timestamp from low-level events, gathering an accurate timestamp is crucial for the application, using low level hooks, makes this much easier. See section 6.1.3 for a discussion about time granularity.

Below is a description of each term:

**Timestamp:** Number of milliseconds since the system was started up.

**Press up / down:** Key pressed / released, or mouse button pressed / released.

**Scancode:** Value that says which key was pressed on the keyboard.

**Virtual key code:** Identifier that Windows uses for identifying keyboard buttons.

**Context code:** Says whether or not the Alt key is held down.

**Transition state:** Says whether the key was pressed or released.

**Extended key:** Says whether it is a function or regular key.

**Injected:** Says whether the key was computer generated or generated by a human.

**Number of keypresses:** If the key is held down, this will say how many is sent to the OS.

**Previous key state:** States whether the key is up or down when the message is received.

**x, y coordinates:** Current placement on the screen.

**Wheel:** Information related to the mouse wheel.

**Handle to window:** Handle to the window that received the event.

When we use low level key or mouse hooks we need to process the message fast enough, otherwise our hook will be removed on Windows 7 and later [17, 18]. How many milliseconds we have to process the message is given in *HKEY\_CURRENT\_USER\Control Panel\Desktop*. On systems earlier than Windows 7, the message is just passed to the next hook.

### 3.1.3 Windows Certification Programme

Microsoft has a program for certifying applications and programs, when an application is certified it is a proof of that it is stable, secure and that it meets the coding standards of Microsoft. To see the entire document with requirements we refer to Microsoft's webpage [20] and to our similar checklist that you can find in Appendix C.

The highlights of this document is that the application should:

- Install and uninstall completely to and from the computer
- Respond to system messages in terms of restarts, system interaction
- Have a valid code signature
- The program should be able to distinguish users
- Be compiled with compiler security settings

## Tools for testing/certification

In the Windows Software Development Kit (SDK) [21] there is a tool called "Windows app certification kit" [22] that analyses application briefly to see if it meets the minimum requirements of the windows certification programme. To accomplish the analysis the tool needs user interaction to install/uninstall the application. Under the analysis the tool observes and analyses the process and behavior of the application being tested.

### 3.1.4 Package management

#### Windows installer

Windows Installer is the native installation and configuration service for the Windows operating system. The Windows Installer is a relational database that contains all the information relevant for an installation – where the most relevant information is:

- Program files and where they should be installed
- Registry entries, the WI can edit the Windows Registry
- Shortcuts
- Merge modules, some programs may rely on DLLs providing special functionality. Merge modules is a DLL package that is compatible with different versions of the Windows operating system

The extension format for the installer is .msi, this naming convention applies to all versions of Windows. [23] [24] [25]. Orca is a tool [26] developed by Microsoft to edit and view MSI-files, by using Orca it is possible to understand and test more specific parts of the Windows Installer if that is needed.

To create a Windows Installer we used the WiX Toolset (Windows Installer XML), WiX is described in the WiX section of our manual in appendix A.

#### Windows installer patch

A Windows Installer Patch is a package that contains the transformation between two program versions. The extension format for a Windows Installer patch is .msp. The advantage of patching is that these files are smaller than the Windows Installer files, because they just contain the bits that are changed in a product. A clear disadvantage is that patches only can change existing files and cannot introduce new files to the installation. To introduce new files and other features you need an upgrade package (a Windows Installer package that removes previous versions and installs a new version from scratch).

#### Certificates

Digital certificates today often use RSA signatures. The owner of a certificate uses the information inside the certificate to create a unique hash that is sent with the signed file. The signature makes it possible for the receiver to verify that the sender is who he says he is. As long as the certificate is keep secure, the user can be assured he knows who created the installer package. [27].

### 3.1.5 Remote logging

One part of the task is to send the logs to a central server. There was really only one requirement, it had to be encrypted. A secondary requirement, implied by other requirements, is that it has to be relatively fast. This leaves us with a lot of different possibilities, the simplest might be to send complete files over the network. This will give us small



overhead, since we can decide ourselves how much data is sent at any given time. It also provides little functionality, since all the data is either in our own format, or in no format. To avoid having to design the format ourselves we used a standard protocol for the transfer, RFC 5424 (Syslog).

### Log transport

RFC (Request For Comments) 5424[28] is the new standard for logging, as of March 2009, it obsoletes RFC 3164[29], which is the BSD (Berkeley Software Distribution) Syslog protocol.

The format for the data is (space inserted for clarity): <PRI>VERSION SP TIMESTAMP HOSTNAME SP APP-NAME SP PROCID SP MSGID [SP SD-ID + (STRUCTURED-DATA\*)] [SP MSG]. Each field means the following:

**PRI** The priority value of the message, can be used set higher priority on some type of messages. This value should be between 0 and 191.

**VERSION** The version number of the protocol, we use version 1, which is the only version.

**TIMESTAMP** Date and time of the message, there are several options for accuracy here, but we include milliseconds, as required. The format is: YYYY-MM-DDTHH:MM:mmmZ, where T is a separator between date and time and Z marks the end of the timestamp.

**HOSTNAME** This is usually an IP-address, Fully Qualified Domain Name (FQDN), or hostname. This is not useful for us, so we use this field for unique ID which we generate ourselves.

**APP-NAME** The name of the application that caused the message. This is only filled if it is a software event, otherwise it is empty.

**PROCID** Should be the process ID of APP-NAME. Is empty unless it is a software event.

**MSGID** An ID to identify the message. We use this for session and event counter.

**SD-ID** This is an just an identifier for the structured data. If we create the ID ourselves, it must contain "@" somewhere inside the text.

**STRUCTURED-DATA** Here we can place data in structured format, each field has the following format: *data="qwerty"*. The data field should be in seven bit ASCII, while the information in quotes should be UTF-8. We can have several of these fields in one message. Using this to send data, gives us more opportunities on the server, when it comes to parsing and storing the data in a readable format. In other words, we are placing the data in context, so any application can read the fields and understand the data, this also conforms to CEE (Common Event Expression)[30] requirements for log transport.

**MSG** The format is UTF-8 or ASCII. Here we can write whatever message we want. If the format is UTF-8, we must start the message with \xEE\xBB\xBF.

**SP** Space (0x32)

So an example of a valid message is:

```
<191>1 2013-05-15T21:22:23.01234Z belt 12 B [b01 log="start"] Started logging
```

## Transport Layer Security

TLS (Transport Layer Security)[31] is a way to provide confidentiality and integrity to data transmitted over an insecure network. We use version 1.1, even though version 1.2[32] is the newest version. The reason for this is that 1.2 is not supported in the libraries we use.

The protocol can be divided into two layers, one is the handshake protocol and the other is the record protocol. The handshake protocol is based on three properties:

1. Each peer can be authenticated.
2. Secure negotiation of a shared secret.
3. The negotiation cannot be altered without detection.

The record protocol serves as encapsulation of higher level protocols, it has the following properties:

1. The connection is private, where the symmetric keys are generated uniquely each time.
2. The connection is reliable, the message must include an integrity check.

## 3.2 Development

### 3.2.1 Documentation

For all projects documentation is absolute necessary for those involved. We have divided 'those involved' into three groups:

**Developers** For developers it is very important to have an in-depth understanding of the application. Proper documentation will help them a long way achieving this. With documentation they can later refer to how the different entities of the application works and operates in conjunction with each other. The documentation for the developer has to be easily maintained, scalable and easy to understand. For software development – call graphs, information and comments about variables, classes, objects and functions is invaluable information.

**Users** For the end user the information in the documentation should be superficial and only tell about the end-user application of the system. telling how to use the application.

**Administrators** need documentation that provides information about how the entire system and is configured, communicates and works in an overall scenario. The documentation should show all necessary information needed for the administrator to maintain the daily operation of the system, configure it and also enough information to identify/mitigate any errors that may occur.

The reason for documenting the source code is both for the current development process and for any future development of the system. Without any documentation about the system no one will be able to understand it in an efficient and pleasant way.

When documenting the sourcecode there are applications that can assist us in creating the document. In Microsoft Visual Studio it is possible to comment in XML[33]. There is also a tool called Doxygen[34] that use certain keywords to index and gather comments in a document, this is one of many open source tools.

### 3.2.2 Continuous Integration

CI(Continuous Integration) is the process of having a project where all participants add their local edits to the a common storage repository as soon as they're made. The common storage repository is usually a version control system like Subversion or Git that provides version control so the edits can be rolled back if needed.

This process results in a project where all participants have the latest version of the project available at all times. Combining this centralized storage environment with a build system provides the possibility to add automated validation checks on new edits that a participant want to add. This could involve running a series of tests on the application, performing a build of the project or another validation check that is necessary. The result of this check can then be used to decide how to handle the new change. The build system may then discard it, allow it, return an error, alert a dedicated person, upload it to a bugtracker, or whatever the build system is configured to do.

The main difference between a version control system and a version control system with continuous integration is small but very significant. A continuous integration system should always contain a running version of the project in the repository. This is because of the build system that will validate each new change to the project before adding it to the repository.

A very big positive mark for CI is that the build system running in the background can be used in many ways. One of them is to use it to show the current state of the application and the result of the tests. This will be able to help with project management, since it provides feedback on the process and current progress.

Though this is a very good system there is also a drawback, but this is questionable. The drawback is that a CI system could potentially fail a participants check-in if it is not compatible with the code that's already been added. Even if the error is in another part of the system. The big advantage on the other hand is that the system should always be in a runnable condition when grabbing it from the the repository. Though it could be errors in the repository if the build system allows the change.

### 3.2.3 Code analysis

Code analysis is the task of going through a systems source code in search of errors, security flaws, design flaws and other errors/flaws that may exists within a system. There are two main methods of doing this, first is code review, second is static analysis. These methods have each their own drawbacks and advantages over the other, but both should be used in every system development project where coding is involved. This will ensure that the system is more secure, more reliable, has fewer code bugs, and has fewer design flaws.

#### Advantages

**Manual Code review** is the task of manually examining a system's source code in order to discover security bugs, design flaws, coding errors and reliability issues. The task of doing this is complicated and very much error prone when doing it without a systematic

approach. The best way to handle a code review is to perform a risk analysis of the system before starting the code review, since the risk analysis will depict in which parts of the system is most critical to review. Another great advantage of doing code review is the possibility to find logical flaws in the system, which automated tools are mostly unable to do.

**Static analysis** was created since manual code review is a tedious and expensive task to perform, one has developed automated tools to analyse the source code for errors. This way of analysing the source code is great significantly reduces the problems that manual code review has. First of all it reduces the time and labor cost of having an employee manually going over the code looking for errors. In addition it makes the development process handle software security issues in a much better capacity, since problems that would otherwise go unnoticed will now be discovered.

Neither manual code review nor static analysis will do a perfect review of the system, even together, but it's necessary to implement both methods in a development process because it will result in a better end product and less refactoring after the end product is delivered.

### **Disadvantages**

**Manual code review** is extremely time consuming and very expensive to perform in a project. Therefore it should only be done in the key stages of the development process, and only when using a risk analysis based approach to decide what to review.

The biggest drawback in this case is the person doing the review. Code review relies entirely on the skills of the person performing the task, since its a manual task. This is also somewhat true for static analysis, but it refers to the one that has to filter through the false positives.

**Static analysis'** biggest drawback is that it is mostly unable to discover logical errors. I.e.  $IF(F > SMAX)$  would be hard for a static analysis application to detect that it instead of  $> =$  should have been  $= =$ . Even though it is possible to discover some logical errors, it is very limited.

Another problem static analysis brings up is the amount of false-positives it yields. This is a big consideration since there may be reported several hundred errors in a an application, but only a fraction of these may be actual errors that cause poses a risk or vulnerability.

The last problem is the ability to correlate the findings within each part of the system to an overall state of the system to analyze communication between the modules. Some tools are able to do this, but this is limited.

### **3.2.4 Bug Tracking**

For all development projects where one have one or more testers, it is a great benefit to use a bug tracking system instead of emails, because emails causes a decentralized environment.

Bug tracking is in its simplicity a system where product testers can submit the errors they have found in a product. Bugtracking software is very useful for keeping a record of how much time that is used for fixing bugs, keeping a record of what needs to be done and also a record of future work. Bug tracking are often available to everybody so that everyone can get a current picture of the development.

## 4 Design

### 4.1 Architectural design

For our architectural design we were set on having a client server architecture from the start because of our projects description. Our project description required that we developed a system where a client based application first ran and stored all data locally to correctly formatted files. Then we would expand this solution to incorporate a centralized storage system which only our employers would have access to.

Since our application is a client based application which incorporates gathering data on the local computer meant our final client server architecture was going to be a thick client which controlled everything, and a server that received and stored the data.

Our application is forced keep the functionality on the local computer since it utilizes functionality that derived from the client. The server is a thin client which performs data storage and data management. The only part of our application that is not on the client is the update service which is partially placed on the server where it keeps the information about our current releases and previous versions updated and available for the client.

Within the client server relationship model from Gartner Group we can see that our system utilizes a very client-centric architecture. BeLT is placed in the "Distributed Data Management" relation where the server handles data management and the client handles all of the functionality, logic and presentation. BeLTs server will manage storage and management of the stored data by converting data between formats. The client will collect all the information, present an interface and data to the user along with storing, formatting and correlating events. Because of this is the data management distributed onto both the client and the server. The servers data management process is to receive the data and store it securely on a centralized platform. Here we store the files generated by Syslog-NG into a directory belonging to the client that sent the data. Later on we have a scheduled task that performs an import of the stored data to the database on the server.

### 4.2 Implementation view

For our development we used a virtual machine running Ubuntu 12.04 server distribution. Because of GUCs network topology we had some problems when creating our log server and update server. This is because GUCs(Gjøvik University College) network topology were blocking standard port numbers, which forced us to use non-standard ports when implementing our server.

Though this was no problem for most of our services it caused a problem for our Bugzilla bugtracking system. Because of this wouldn't sendmail work, to fix this we implemented a perl module that enabled us to use Gmail and a Gmail account to send/receive mail instead. During our entire development process we used the same virtual machine for our update server and our logging server, but in future implementations it should be implemented as two separate hosts to avoid any performance issues.

Since we are collecting and transmitting highly personal and sensitive information across public and insecure networks we had to implement countermeasures to ensure

the confidentiality of our data. To do this we generated certificates and applied for our own server certificates which is trusted by TERENA. With these certificates, based on a 2048b key, we implemented TLS encryption on all traffic between our server and the clients. We also implemented encryption between the update server and the users using the HTTPS protocol.

The data transmitted between the client to the log server is a TLS encrypted data packet. Inside this packet we have formatted our data according to the Syslog protocol in RFC5424. This way it can safely traverse the Internet without the data being read by a third party. This communication channel is mainly a one way communication line, from the user to the logging server. The only communication transmitted back to the user is standard TCP and TLS communication to maintain the connection. When the server receives the communication from the client it performs the stages as explained in section 4.2.3.

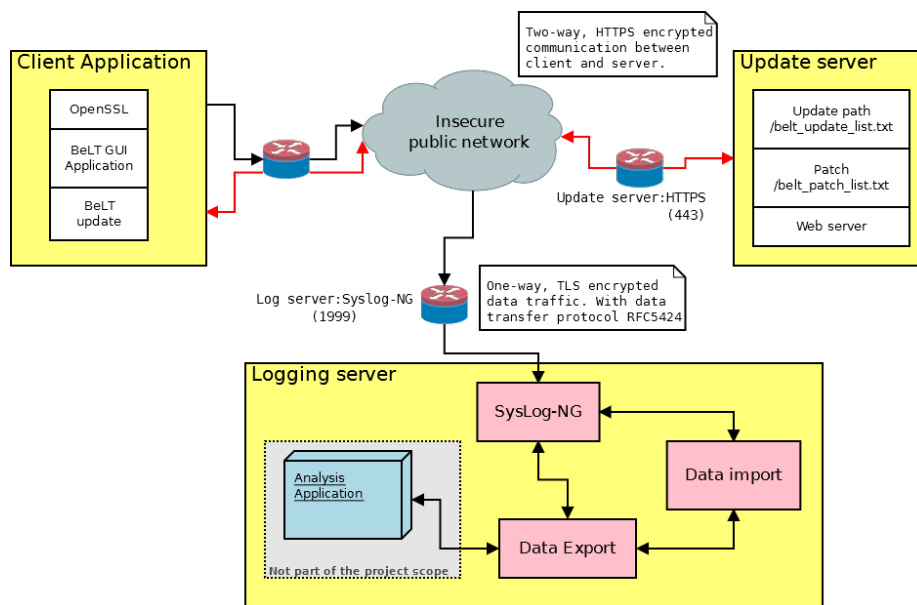


Figure 1: Implementation scheme of system architecture

#### 4.2.1 Client application

The client application builds on several aspects and functionality. We are capturing mouse and keyboard interaction by using a Windows hooking functionality. To capture software events we are using the functionality that resides within UI Automation, see 5.1.1.

Because we want the user to be able to control BeLT we have given our GUI(Graphical User Interface) the ability to control the application. The GUI will start, stop, pause and resume the logging functionality and process the data according to its settings configuration. As a part of our application we have implemented TLS encryption on our communication between the client and server. To add this functionality we have use the OpenSSL API to create, initiate and maintain our connection to the server.

During our development process we created a start-up sequence of BeLT which would access a file on our update server which would tell what the newest version number is. Based on the retrieved information BeLT would either download and update itself or

skip the update sequence. The update procedure is further described in section 4.2.2, but since our project won't be regularly maintained after our projects final release, the update functionality will be disabled.

#### 4.2.2 Update server

For BeLT to update itself, it has to communicate with a server that maintains information about which releases that has occurred in the past and what the current version is. Our update server maintains two separate lists, first is the file containing the version number for the latest release. The second is a list over all patches that has occurred.

BeLT first reads the current version number from the list, then checks to see if its own version number matches. If it matches BeLT will continue as normal, but if it has a version lower than the latest version BeLT will continue by reading the second file. This list contains the patch history which means that it will have to read the file until it finds the version it is currently on. When this happens it will download, either the 32-bit or 64-bit executable and execute the installer, thus initiating the installation process without any user interaction. During this update the user should not be bothered and it should happen with as little interaction as possible.

To manage access to these files our server runs a Apache2 webserver where we've added these two files in the root of the belt servers web area.

#### 4.2.3 Logging server

Our logging server is a standard Ubuntu server running Apache2 as the webserver, MySQL as the database service and Syslog-NG for our logging service, see figure 3 for the logical view of the server.

First and foremost it runs the Syslog-NG server that receives and stores the data from the client application. The server receives the encrypted data packets from the clients and decrypts it using the issued certificates. Then Syslog-NG creates a new file or appends the data to an existing file using the predefined file format in the Syslog-NG configuration file.

Then it in addition to the Syslog-NG service it runs a scheduled Cron task which executes our Data import script which inserts data to the MySQL database at specific times. Then it also contains the data export script that export data into a CSV formatted file for our employer. The logical view of the server is explained in in section 4.3.2.

### 4.3 Logical view

#### 4.3.1 Client

Figure 2 shows a logical view of the client, the yellow boxes indicate components and arrows indicate information flow.

As you can see from the figure, information flows mostly in one direction, the **graphical user interface** sends events to **data capturing** and **update service**. The update service runs by itself and update if needed. The data capturing part runs by itself and sends any data it collects to **data processing**. Data processing can send data back to the user interface and to the **transmission** component. The transmission component is responsible for sending all the logs to a central server.

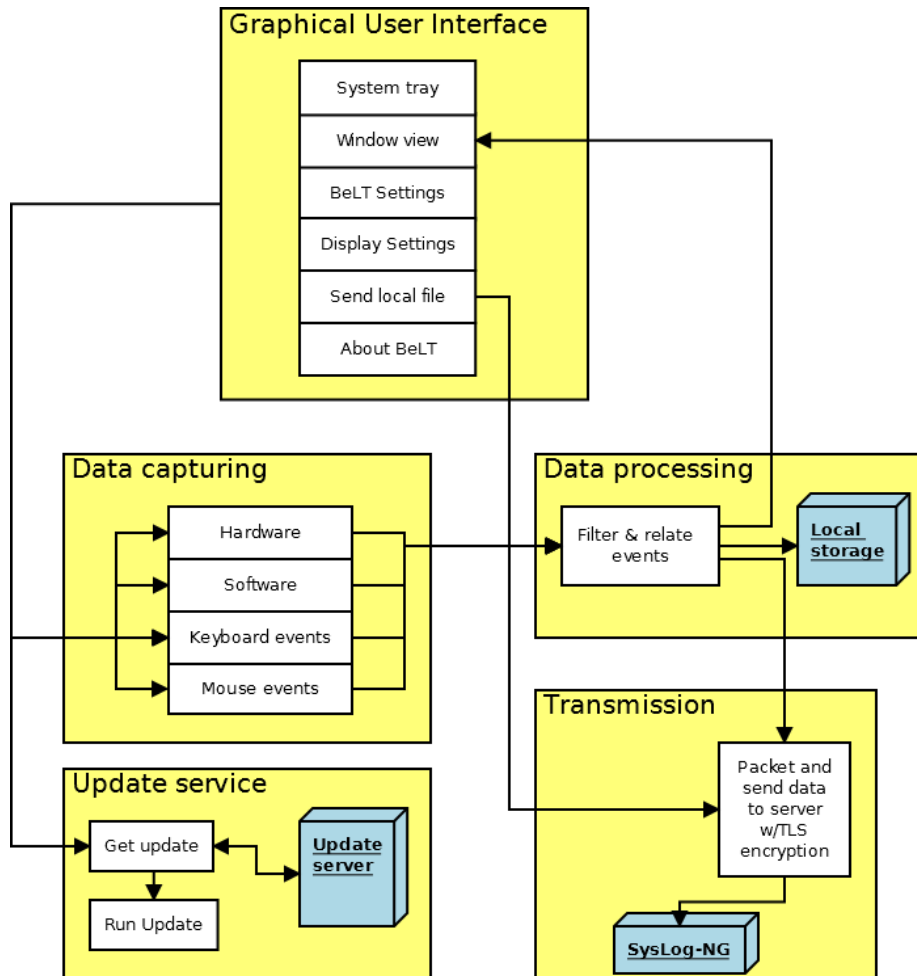


Figure 2: Logical view of the client application

### Graphical User Interface

This is the view presented to the user. The program is supposed to run in the background, with very little user interaction, so the design is very simple. We have configured some simple option that can be set by the user, but this is not necessary, it is only there so the user can see what is going on. The user also has the option of storing data locally before sending it through the server. The user can then decide if the data can be sent to the server or, if it should be deleted or if certain timeframes should be excluded.

The GUI consist of six sub-components:

**System tray:** This is meant as the real display for the user. The user shouldn't have to see anything more than this icon, unless the user want to. This icon will change colors when the application is stopped, started, paused or have detected a password field.

**Window view:** This dialog is hidden by default, but the user can display it to show what is going on.

**BeLT settings:** Main settings are what kind of storage to have, can be one of three options, local CSV files, local network files that can later be sent to server, and send concurrently to the server.



**Display settings:** If the user want to see what is logged, in real time, here he can chose what to display. Nothing is displayed by default.

**Send local file to server:** We keep track of all network files that has been stored locally. In this dialog, the user has the option to delete these files and send them to the server, optionally filter out certain timeframes.

**About BeLT:** This dialog provides some minor information about BeLT. It states some information about BeLTs purpose and functionality and a required text to comply with our use of OpenSSL.

Both the system tray and the main dialog serves as entry point to the main functionality, like start, stop, pause and resume.

### **Data capturing**

This is where we collect all the necessary data. It runs on six separate threads, but can be divided into four logical components:

**Keyboard interaction** We use a Windows hook to gather information about how the user uses the keyboard. See table 8 for information about what type of information we store.

**Mouse interaction** We use a Windows hook to gather information about how the user interact with the mouse. We have also implemented mouse compression in this part, for a full discussion on the mouse compression, see 5.4.1. For a full overview over what type of information we store, see table 6.

**Software interaction** This is where we try to find out what happened when the user pressed a key, moved the mouse, or something similar. We monitor all the applications for certain type of events. Whenever we receive an event we send it to processing. See table 7 for a full overview of what we capture.

**Hardware** Here we gather some basic information about hardware that helps put the remaining data in context, like screen size, mouse hardware, keyboard type. We also gather average CPU and RAM usage. See table 9 for a full overview of what we store.

Every single event has a timestamp attached to it, but the hardware and software timestamp are generated by us, and we have not taken any precautions to make them accurate. The timestamps in mouse and key events are given to us in the Windows function and should have millisecond accuracy. See 6.1.3 for a full discussion on time granularity.

### **Data processing**

Every single event that is registered is immediately sent to the data processing module. The tasks for this module is to relate events, filter out unnecessary data and format the data according to the format of the Syslog protocol.

The data processing module will when receiving events filter out unnecessary events. This is because certain software events can generate several, equal events which we will receive.

All mouse and keyboard events can be seen as input from the user and software events can be seen as a consequence of that input. For later analysis it can be useful to know

which events is connected with another events. Inside the data processing module we try to find out how these events relate to each other. For a full discussion on how we do this, see 5.4.2.

The data processing module receives all its data as a structure containing event specific information. The data processing module is then responsible for correctly format the data as described by the Syslog protocol(RFC5424). Because of this the data processing module has to ensure that the formatted data is valid and understandable by the receiving Syslog-NG server.

When the data processing module has finished processing an event, it is added to our list in memory. When that list has reached a predetermined value<sup>1</sup>, we create a new thread and send them all to the the transmission module. Each event is still sent separately, but we are gathering all of the events stored in memory to avoid a delay, on each event

### Transmission

This module has the responsibility to set up an encrypted session between the client and the server and send all the data to the server.

If we are unable to connect to the server, or we are unable to send an event, we will keep it in memory. When we have reached a certain number of events, and we still have not gotten a connection with the server, we will store those events to file, so they can be sent the next time BeLT is started.

### Update service

This module has the responsibility to make sure that the user is always running the newest version of the application. It checks against the server if this is the newest version, if not it downloads the new version and installs it. The update server is just a web server with a CSV file with enough information so we can check if we have the newest version and download new version if we need to. Since the application will not be in active development when we finish the project, this part will be disabled.

### 4.3.2 Server

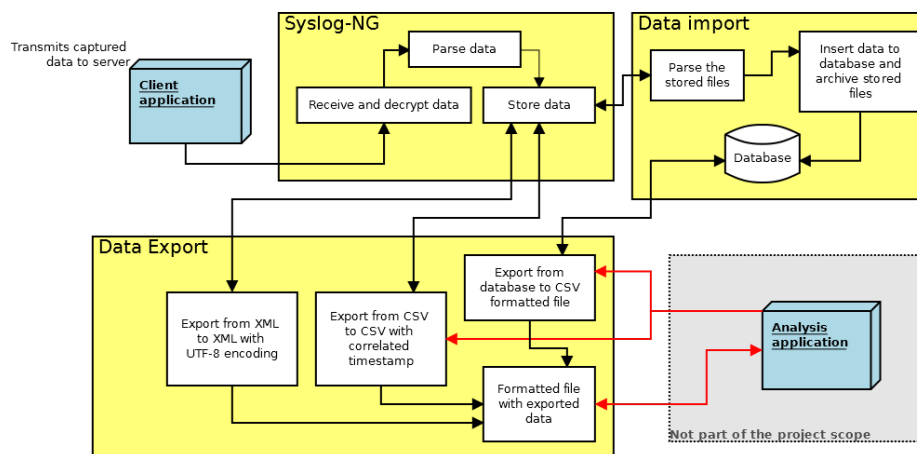


Figure 3: Logical view of the server application

<sup>1</sup>In our implementation it was 500 events

Figure 3 shows the logical view of the server application. It can roughly be divided into three parts, **Syslog-NG**, **Data Import** and **Data Export**.

### **Syslog-NG**

We use an already existing product to receive, decrypt, parse and store the information within the events we receive from the client. Our task in this is to find a productive way to format, structure and store the data. This is to make it possible for our employer to retrieve the stored information and perform the tasks he needs to perform.

To achieve this the stored data should be understandable the way it is stored, it should have low overhead and a satisfactory low storage size. Because Syslog-NG implements the Syslog protocol means we have to abide to their protocol format. This means that all of our traffic has to follow their specific, and more or less fixed format. As a result we are unable to retrieve our information correctly formatted directly from Syslog-NG which is why we have implemented additional methods for processing the stored files to generate our own correct file formats.

We are able to store the information received from clients to both CSV and XML formatted files with Syslog-NG itself. The XML formatted files are used as input to the data export method when importing data into the database using a scheduled task that imports the data into a database. The CSV formatted files on the other hand is stored and then parsed using a second export script that reads its information and converts all of the events timestamps to a integer value displaying the amount of milliseconds since the start of the users session.

### **Data Import**

Because we wanted different formats we also had to be able to convert the formats into the format used by our employer. One of the formats we wanted to use since the dataset will become very large and may become slow to when performing searches. Since a database is a very fast method of accessing specific data within a large set we decided to implement this feature for future use, even though it wasn't a requirement.

To import data it is necessary to store the data as XML. Because of this have we made the import script perform two tasks dependant on what the user wants.

1. Transform Syslog-NG XML into well formed XML with UTF-8 encoding(See next section).
2. Take the well formed XML and insert it into an indexed database.

By importing the data to the database are we making it easier to create and manage a large data set. This will make it a lot easier to perform statistical information gathering and correlation, i.e find out what a users most typed button is or his most used application.

More information about the script is described in the "Database" section of our system manual, Appendix A.

### **Data Export**

We have created three scripts that export and converts the stored data into a more usable format than it's original. Our first script transforms data from the CSV-format stored by Syslog-NG into a CSV-formatted file. This is because the files generated by Syslog-NG contains timestamp values in the format "2013-05-15T12:00:00.0123+00:00". The timestamps has to be changed to a numerical value that represents the time in millisec-

onds after the the session has started – since it easier to analyse for our employer.

The second script performs the task of converting our information stored in the database to a CSV formatted file with correct timestamp values. This script will retrieve the data from the users sessions, either by retrieving all or specifying which users and sessions to export.

We export to CSV because this is the format that is our employer will use during his analysis. After some time it might be more efficient to export data from the database storage – because it can be indexed for faster retrieval of specific elements. See the system manual(Appendix A), to see how we have used the indexes.

The third script is actually the first sequence of the import script. This script requires that Syslog-NG have store the data in the XML format. This script will convert the files generated by Syslog-NG into a new XML file with the same data, but it will convert it to UTF-8 encoding before trying to escape any unwanted characters like "&" by replacing it the escaped character symbol "&". This is because an XML file with invalid characters stored will not be well formed, and by converting the file along with escaping certain characters we ensure that our files are well formed.

## 4.4 GUI design

Since our application is running unobtrusively on the computer, meaning it should never get in the way of the users actual tasks. Because of this our main user interface is the BeLT icon located in the Windows system tray. This icon provides subtle information about BeLT's current state and give the user information about its presence. We also created a full window interface, because we wanted a user interface that could provide the user with more option and show in real time, what information is logged. In the following sections we'll introduce the initial design and the final GUI of BeLT

Since a single icon is not satisfactory as a UI(User Interface), for the user, we have extended it by adding an additional view for it. We have implemented a dialog that displays our "window view" for BeLT, a larger interface with the ability to display captured data in real time. It also allows for configuring some of BeLTs functionality.

### 4.4.1 System tray

Our initial design and final implementation for the system tray is quite similar, but with a reduced number of items in the menu when right clicking the icon as seen in figure 5 as opposed to 4. We have kept the icons in the system tray unchanged, but removed several buttons to let the user have an easier experience when using BeLT. In the final version it has one button than interactively changes according to BeLTs current state. If the application is stopped the button will read "Start", if BeLT is running "Pause", if paused from running it will read "Resume". However the "Stop" button will remain the same under any scenario because the user must be able to stop logging completely at any time.

Our menu also contain the "Restore"/"Hide" item which based on whether BeLT is currently hidden or shown. It will hide the application from the desktop and only show BeLT through the icon in the system tray if BeLT is currently shown or restore it to full view if it was hidden.

Both our initial and final design have four separate indicators that informs the user about BeLT's current state, see table 2. The red icon indicates that the capturing is turned

all the way off, the blue icon indicates that BeLT is currently paused from logging by only halted and not stopped. Green icon indicates that BeLT is currently running and capturing events per normal behavior. The final and yellow icon indicates that BeLT has discovered a password field and will censor all keypresses until the the user has left the password field.

Icon status indicators		
Color	Sign	Indicates
Green	"B"	BeLT runs normally without problems
Blue	"B"	BeLT is currently paused
Red	"B"	BeLT is currently stopped
Yellow	"!"	BeLT has discovered a password field

Table 2: Status indicators for system tray

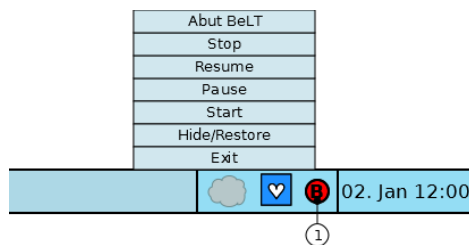


Figure 4: Initial system tray design

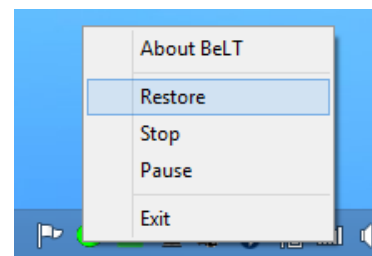


Figure 5: Finished system tray design

#### 4.4.2 Window view

For our "Window view" we needed two things; an area to display the captured information and control buttons for start, stop, pause and resume the logging. Even though BeLT is supposed to run hidden, we wanted to have a GUI that would help us with user acceptance since we are logging very sensitive information. We have illustrated our initial window view and our final design in the figures below.

The window view was redefined quite a lot. We started out wanting to list all the processes logged in a field on the left side, mark 4 in figure 6. This would allow the user to filter based on processes, but was abandoned because it would take too much time to implement. We also abandoned it because our employer didn't think it was necessary. We instead chose to have one large information field inside the window where BeLT display the data we've captured based on the users preferences. The display field show in the final view the same as the field as indicated mark 3 of figure 6.

On our initial design we wanted four fields in the bottom of the application, mark 5, 6, 7 and 8. These would display different statistical data about the users actions since the application was started. In our final version we implemented two fields in the top right corner of our application which displays number of mouse and keyboard presses.

Since we abandoned our idea of process filtering, we chose to implement a filtering mechanism that allows the user to select what keyboard and mouse events to display in the information field. It is similar to the idea we had behind our two buttons(mark 1 and 2) in the initial GUI. The difference between this was that our initial mouse and keyboard filtering was connected to the process filtering. This means they would filter keyboard

event from the process chosen and either show or not show keyboard events or mouse events. What we did in the final version is that the display field only shows keyboard and mouse events. These events are also possible to filter in more specific manner by using the "Display Settings" dialog.

As seen on our initial GUI we had four different control buttons which also here are implemented as only two which interactively changes its text and functionality. To maintain consistency with other Windows applications we've added all interactive elements to our File menu, Settings menu and Help menu where we've implemented the interactive fields and opening of dialogs. These additional dialogs are shown and explained in the sections below since they weren't any part of our initial design. They were implemented out of need and functionality/requirements added throughout the development.

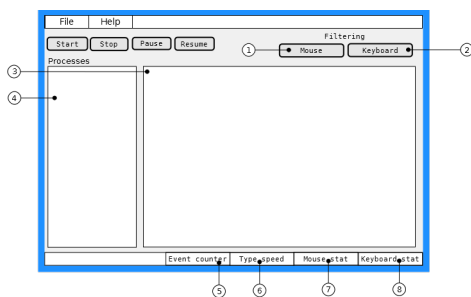


Figure 6: Initial application GUI design

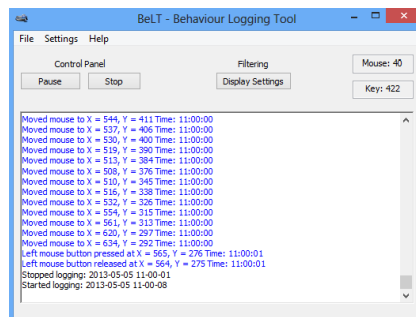


Figure 7: Final window view design

#### 4.4.3 Additional dialogs

##### BeLT settings dialog

During our development process we had to initially store the data on the local machine which we never removed because it became a necessary feature both for our self, our employer and the project. This helps us with upholding our need to maintain transparency of what BeLT is collecting of information from the user. Later in the development process we implemented the functionality to transmit the collected data to a centralized server.

So to let the user decide how to store their data, and set certain system settings we created the "Settings" dialog, figure 8. This dialog is split into two modes, first we have the basic mode which lets the user chose to start BeLT automatically when the user logs on. This feature is by default set to on, so the first time the program is run it will try to create the shortcut. We have implemented this by creating a shortcut to our own application within the "Startup" folder within the windows start bar menu. Secondly it allows the user to chose if BeLT should look for updates on startup by connecting to a specified server, explained later, which is by default enabled.

The next option is an array of radio buttons which gives the user the ability to chose how to store the collected information. This is the default method used to store the collected information. Option number two is to temporary store the files locally until the user itself chooses to manually transmit the files using the "Send local file" dialog, see section 4.4.3.

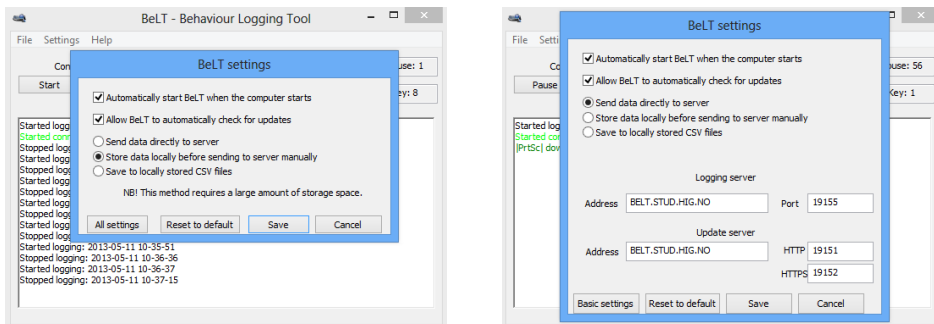
If option number two is to be chosen a warning will appear below the array of radio buttons, which informs the user that the chosen storage method will consume a large amount of storage space.

The third option is to store the files as CSV formatted files. This option will render the files unable to be transmitted to the server unless it is manually transmitted. This is the end of the basic settings, by clicking the "All settings" button the dialog will automatically expand and display additional settings.

First of all the settings for the "Logging server" which consist of the the "Address" field which must be either a valid IP-address or a domain name for the server. The "Port" field must contain the port number on the server that the Syslog-NG daemon listens to for TCP communication.

The following three input fields is for the update service. This information specifies the IP address or domain name of the server update server. In addition it also specifies both which port numbers the update server utilizes for HTTP and HTTPS communication.

The button "Reset to default" will reset all fields to their default settings and automatically save the settings, the "Save" button will save the current configuration and exit the dialog. The "Cancel" button however quits the dialog without saving any changes. By saving the settings we ensure that the users customizations is permanent until changed again.



Basic settings

All settings

Figure 8: BeLT GUI: Settings dialog(Basic/All settings)

### Display settings dialog

Because of our implementation of a display field inside the "window view" interface we wanted to let the user be able to choose for themselves what type of information BeLT would display.

The "Display settings" dialog, figure 9, utilizes an array of checkboxes that is interactive in the way that it will automatically checkmark the checkboxes connected to it. I.e checking the "All Key Interaction" box will automatically cause the checkboxes "Key Released" and "Key Pressed" to be come checked and uncheck them if the checkmark was removed. The same is the case with "All mouse interaction" and "Mouse Buttons" which will check or uncheck all of its child elements when interacted with.

In addition one has the option to check or unheck all boxes. Even though we capture much more information than just keyboard interaction and mouse interaction, we have chosen to display only these two types of events. This is because the data gathered from software events with UI Automation is not user friendly to view in any form and is not easily understood. On the basis of this we have only implemented the methods for viewing keyboard and mouse events for the public release version of BeLT, even though we

have the functionality in place in use for development purposes.

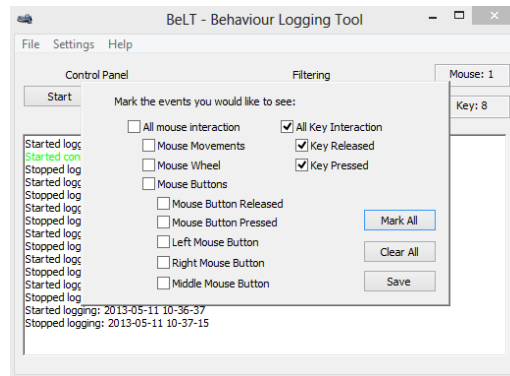


Figure 9: BeLT GUI: Display settings dialog

### Send local file dialog

As part of our BeLT settings the user can choose to store their files locally before sending them to the server manually. This dialog, figure 10, was designed and created to manage this possibility. First of all, the left field, "List of files", displays all locally stored files that has not yet been sent to the server and displays them by identifying them with the first and the last timestamp of the session.

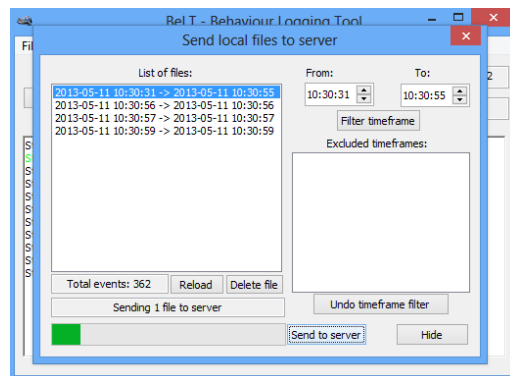


Figure 10: BeLT GUI: Send local file dialog

Below the file viewer we have a information field and two buttons, the information field displays how many events the file contains. The "Reload" button refreshes the file viewer and append files that been created since BeLT first was loaded. The "Delete file" deletes the marked file in the file viewer from the computer.

The "From" and "To" timers on the top right is used to select a specific time interval that the user can exclude from the locally stored file before sending it. This is very useful if you i.e, know when you were performing a banking transaction that you wish to avoid transmitting. Then you can specify the time interval and click the "Filter timeframe" to exclude it from the file when sending it. The display field below is automatically updated when one adds a specific timestamp, and lists all currently excluded timeframes. The button "Undo timeframe filter" will remove the currently selected timeframe within the list of excluded timeframes.



The final two buttons are the "Send to server" which takes the currently selected file in the file list and transmits it to the server and displays the progress with the progress bare on the bottom left. While this dialog is running and the user is transmitting files the "Hide" button will hide the entire dialog to allow for the transmission of the file to run in the background undisturbed.

### About BeLT dialog

We implemented the standard "About" dialog, figure 11, to BeLT in order to comply with our use of the OpenSSL library. In addition it also gives us the opportunity to inform the user of what BeLT is and what BeLT does. We've implemented a very short description about BeLT here along with the current version number that is running. In addition to stating what version is running, we remind the user that BeLT will capture and log sensitive information about the user and store it on a centralized server.

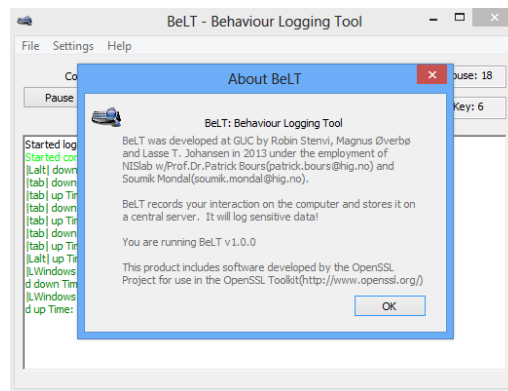


Figure 11: BeLT GUI: About BeLT dialog



## 5 Implementation

### 5.1 Application

#### 5.1.1 User Interface Automation

Because of the way this API works, we need to gather a lot of information, typically every time an event occurs. To make this work fast, Microsoft has implemented caching[35], with this we can save a lot of data while events occur and then retrieve it immediately. This is an important part to make the program efficient.

Since we are retrieving events on the entire desktop, including our own application, we need to make all UIA calls on a separate thread[36]. This thread need to be a non-UI thread, which means that it should not have any UI elements, in practical terms, it means that no user will ever interact with it, so we remove the threading issue. If this is not done, the application can run slowly and in some cases, stop responding. What complicates this further is that, when removing the event handlers, it need to be done on the same thread that created it.

#### Application support

We can only retrieve detailed information from applications that support UI Automation. But even if the application doesn't support UIA, we can still retrieve some information. We get events like pressing a button and opening of menu, regardless. The name will then be what it says on the button or menu. We get other events also, but they may not have a name, so we can't necessary understand what is going on.

But we may be able to correlate, since they have an ID, called the "AutomationID"[37] that is unique among it's siblings. The fact that it is not unique on the entire desktop it's not a big problem, since we can check between process, element type and element tree. This identifier should be the same every time it's run, but not necessarily the same between different versions of the program.

The following is a list of applications that support UIA, it is not complete, as we only check applications that we are interested in and we have available<sup>1</sup>. The tests are a basic run-through of what we consider normal usage.

List over applications tested with UIA	
Application	Status
Microsoft Word, Excel and Outlook 2013	We receive the events we are looking for, but the names are not always intuitive, especially when navigating through menu-bars. Some events like zooming is only received when using the buttons, most events are received, regardless of how it's accessed.
Visual Studio 2012 Professional	Missing some events when using the editor
<i>Continues on the next page</i>	

<sup>1</sup>The version numbers only indicates which versions we tested. We have no reason to believe that earlier or later version are not supported

Application	Status
Firefox 18	We get all the regular events, but we don't any events that happen inside the web-page, like clicking on a link.
Internet Explorer 10	Same as Firefox
Opera 12.15	We don't receive events about switching between tabs, and we are unable to detect password fields inside web pages.
Google Chrome 26	We get most events, but we are missing names for most elements, and we don't detect if there is a password field or not.
Metro applications	From our perspective, these application work like any other applications, we can detect when they are used and how they are used.
Thunderbird 17	We receive most events, except for menu interaction.
Skype Desktop version 6.3	We get the most basic information about menu-changes and we can understand when the person is talking and using Skype. But we are missing a lot of names for various elements.
Notepad	We seem to get everything we where trying to receive.
PowerShell ISE	With regular PowerShell, we get the same as we do with CMD. The ISE version has a much more graphical interface, where we can retrieve elements just like any other application.
CMD	We can retrieve all elements beside the text window.
File explorer	Some small issues with names, but other than that we get a pretty complete picture.
Putty	Same as with cmd, we get everything beside the text window.
Adobe Reader XI	We receive almost all information, but not scrolling, except when the page is changed, we are also missing some names.

Table 3: List of application tested with UIA, and our experiences

The main problem with using UIA is that there is no guarantee that events are raised when something happens. This can be a problem regardless of whether UIA is supported or not. This is something that has to be considered in the analysis phase.

### Password

We are interested in excluding any sensitive information from our logs, both for privacy reasons and ethical reasons. In most cases it is hard to see what is sensitive information, but passwords is one type of information we are able to detect and exclude from our logs.

All edit fields in UIA has a boolean value[38], that says whether it is a password field or not. This value also says that keypresses should now be echoed back to the user. If

we don't print the keypresses we see inside these boxes, we should be able to exclude passwords in our logs.

It is important that the user knows when we detect and not detect a password field, so whenever we detect a password field, we change to a yellow icon. That way the user knows that we don't store those keypresses.

It is important to notice that this only apply to GUI edit fields, we are unable to detect passwords typed in terminal based applications. We don't expect this to be a big problem since most users will probably use graphical applications. An example of an application where this is the case is "Putty", when dropping to a terminal, before authenticating.

As table 3 also shows, we are not able to detect password fields in all browsers. This problem might be solved with polling, and/or looking at the label for each field.

### Logged events

Table 4 shows all the software events we gather, their corresponding constant in the Windows library and our description of when they happen.

List over events gathered with UIA		
Name	Windows constant	Description
Visual change (VC)	UIA_WindowWindowVisualStatePropertyId	If the the window is resized. This can be manually by user, maximize, minimize or re-stored from minimize.
Focus Change (FC)	N/A	Whenever the element in focus has changed, this can either be the top window or any element inside the main window.
Window Opened (WO)	UIA_Window_WindowOpenedEventId	Whenever the user starts a new program that opens a new window, this event is raised.
Element Invoked (EI)	UIA_Invoke_InvokedEventId	Typically pressing a button.
Menu Mode Started (MMS)	UIA_MenuModeStartEventId	This happens the first time the user clicks on a menu.
Menu Opened (MO)	UIA_MenuOpenedEventId	This event is raised when moving between menu items.
Text Changed (TC)	UIA_Text_TextChangedEventId	This event is raised whenever the text is changed in some user editable element.
Object Change State (OCS)	EVENT_OBJECT_STATECHANGE	Part of Microsoft Active Accessibility (MSAA) and not UIA, but contains some events we are unable to gather in UIA. These events are things like changing the state of an element, like pressing bold in Word for example.

Table 4: List of events gathered with UIA

MMS and FC will not only indicate what and how the user did something, but it will also say in what state the user is in. It will for example say if the user is in a document, an edit field or if the user is navigating a menu. FC alone with key interaction and mouse interaction, might be enough to draw some abstract conclusions about the user, while the remaining events can maybe say something more concrete about exactly what the user is doing.

## Potential problems

Some applications, like Adobe Reader will notice that accessibility application is running. Adobe Reader will then try to get the user to set up Adobe Reader, so that it can run better. This is not a major problem, but can cause some nuisance for the user. We have documented this in our user manual, see appendix B.

### 5.1.2 Package management

#### WiX

To create Windows Installer packages we use WiX Toolset (Windows Installer XML toolset) version 3.7. WiX has the same functionality as the Windows Installer itself, and it also supports the creation of databases and making rules in the Windows firewall.

WiX configuration is written in XML and it can be built from a commandline environment. For us the support for commandline tools was very important, because we wanted to script the whole process from compiling binaries, codesigning these binaries and building a windows installer. By scripting the whole process we know that our product always will be consistent. We know that humans make mistakes and that computers don't. The process of creating the scripts took a lot of time, compared to just compile the different parts on demand. We can justify this choice because we know that without us doing this, there would be a lot of confusion for those who will continue our project. We have described how WiX works and how the scripts work in the manual.

At the early stages of our project we used Innosetup [39], it doesn't support Windows Installer packaging, but when we started to work in January we didn't have enough knowledge about package management to make a good choice on what to go for. We knew about WiX and Innosetup, and WiX seemed too complicated to use compared to Innosetup. After two weeks of using Innosetup and reading about the world of installers, we decided to use WiX instead. The key factors that made us change our mind was that Innosetup didn't support Windows Installer format, merge modules and patching. Our WiX-based installer and the scripts surrounding it was developed throughout the whole project, but now it supports the necessary functionality.

#### *Patch*

We programmed a solution inside the BeLT to download and run patches. The automatic update retrieves a patch-file from the server (encrypted session), checks the certificate and applies the patch with a tool `msiexec.exe` (Microsoft native tool). How we made the patch-file you can read about in the system manual (Appendix A).

#### *Code signing*

Certificates are based on a Public Key Infrastructure with trust. There is a root authority that verifies publishers, and this way makes a chain. When Windows verifies a certificate, it does not only check if the instance it got it from is legitimate, but also if everybody in the chain is legitimate too. We have codesigned our application and binaries with certificates from TERENA[40]. TERENA is an institution that provides human networking and collaboration in Europe, and they are also a certificate authority. We have written more about the practical side of codesigning in the system manual(Appendix A).

### 5.1.3 Remote logging

We chose the "Syslog-NG (Next Generation) OSE (Open Source Edition) 3.3 LTS (Long Term Support)"[41] as the service to accept the logs. Our client application sends the logs

according to RFC 5424[28]. We use OpenSSL as a wrapper to encrypt our communication[42].

### Syslog-NG

As said before, we chose "Syslog-NG OSE 3.3 LTS", OSE, is the open source edition of the premium edition. This supports UDP, TCP and TLS. TLS is a requirement for us, since we are collecting and processing sensitive data.

The premium edition of "Syslog-NG" also supports encryption when storing the data, which is desirable in a production environment. It also provides more opportunities as to where the server is kept.

### OpenSSL

We chose to use OpenSSL instead of Microsoft's standard libraries for TLS, this was because we used a Linux server which ran Syslog-NG and we wanted to avoid compatibility issues. The library imposed very few restrictions on our application. All we had to do was to place a notification, that stated our code contains software developed by the OpenSSL Project[43].

It also supports mutual authentication, which is a good security mechanism on the client. This way we made sure that no one was able to impersonate the server. The server can also authenticate the client, but that is not important for us, but in a more closed environment, this can be used to limit who can send data to Syslog-NG.

## 5.2 Server

As part of the project we had to implement a centralised storage facility. We implemented this by installing Syslog-NG on a Ubuntu 12.04 server on a virtual machine at GUC(Gjøvik University College). With Syslog-NG, we could use our server as a centralised storage facility for our application that can output our data in the following formats.

- RAW<sup>2</sup>
- XML
- Indexed database
- CSV
- Unindexed database

There are pros and cons to all of these file formats, but some are worse than others when it comes to scalability, future implementations and performance. During our testing, section 6.3, of the file formats we found out that continuously retrieving and inserting data to an indexed database is by far the worst format to use.

First and foremost we wanted to have a baseline test (RAW format) where we wrote the received data from the users directly to file without parsing or manipulation of the received data. This format is not very usable without a lot of extra work and can't be used for analysis, or be displayed in a good way. We've only used this format for creating a baseline when testing our server.

CSV formatted files, Comma Separated Values, involves separating each value received from the user is separated by a comma. In our file we escape "," with "%2C" (it's hexadecimal representation), all other values are written as is. This ensures that there is no confusion of where each field begins and end. The biggest disadvantage with this file format is that, you have to understand what each field means to analyse the data. The big pro of this file format is that it is very easily read, implemented and generated. It doesn't require much resources, storage space or any additional software to read or write it.

<sup>2</sup>No filtering or manipulation of data, every event is dumped to file.

XML files biggest drawback is that it requires a lot of padding around its values for it to be readable. Each value has to be encapsulated by a fixed tag inside "<>" brackets, i.e. for our numeric flag values we have to store them as "<flag>1</flag>". This generates a need for storage space which is wasted on padding for the actual values. Another problem with XML is that there are symbols that are not allowed to use, adding these symbols results in a corrupted file format. However, this error is easily fixed by escaping these non-valid symbols. I.e. the "&" symbol is not allowed in XML, but by replacing it with its escaped value "&amp;", we are able to avoid corrupting the file format.

Though there are a few serious drawbacks with XML, it is still a very user friendly file format. Because of the padding that contains specific names, we can easily read this file by querying for the value within a specific tag. Which as opposed to CSV does not necessarily have to exist and thereby making it easier to read and use later on. A very powerful and nice feature is that XML files can be processed using XPath expressions and XMLTransformation. Where XPath expressions find and select values within the XML file while XMLTransformation utilizes a stylesheet to transform the entire document into another format.

Though the Syslog-NG configuration to generate XML configuration is complex and long it is no problem and offers no performance issues, just as CSV, when parsing the received event.

The database format we used was a MYISAM database, which we chose to use because we needed to have a database without any indexes or keys. The unindexed database inserts the data to the database without having to create an index for each field that has an index related to it. This file format requires less storage space than an indexed database since each index creates a relation that requires additional storage space to be represented. This means that for each event received the database had to create several additional indexes for the event that had to be created and added to existing indexes.

Because of this it requires much more resources to add events to an indexed database, than an unindexed database. The pros of having an indexed database is that it is much faster to search through an indexed database than an unindexed. This is because the indexes are ordered based on their value and has a relation to its corresponding database field.

The most important drawback with Syslog-NG and databases is that Syslog-NG doesn't have an internal method to import data to the database. This then forces one to implement this file format using external scripts and tools which makes it very time consuming.

### 5.2.1 CSV file format

In addition to storing our data as XML and in relational databases we have also created a CSV file format. We've done this because it was the preferred file format to our employer.

Since we have several different event types mouse, keyboard, software and hardware we had a problem with CSV since these events don't contain the same amount of information.

Because of this we have developed a file format where the first three values are the same throughout the format and based on these three values one will know which fields come next. All events are identified by the following fields:

**Event ID** is an integer that represents the order of when the event occurred. Here shown



by  $n - 1, 2, 3 \dots n$ .

**Event type** is a letter that represents the type of event that occurred. B - BeLT, S - Software, K - Keyboard, M - Mouse and H - Hardware

**Action** is a specific action or type within the *Event type*. These codes are described in detail under each table for their event type. This further divide what type of event that happened.

Other values that are presented in the same way throughout a line is:

**Time** is the timestamp in milliseconds, written in ISO8601 compatible format, when written directly from Syslog-NG. If exported to CSV, either from the database or from CSV. The timestamps from all events, except BeLT messages will be an integer representing milliseconds since the start event (first line). An example for timestamp in ISO format is: *2013-05-15T12:13:14.0123+00:00*. This includes full date, time with milliseconds and time zone.

**Relation** Says which event this is related to, points to an *Event ID*. See section 5.4.2 for a discussion of relations between events.

In the tables below, all fields that are printed directly, are marked in bold, if the text is not in bold, it should be replaced by something else, which is described in the list above or below the table.

BeLT system-events			
Event ID	Event Type	Action	Time
n	<b>B</b>	<b>start</b>	T
n	<b>B</b>	<b>pause</b>	T
n	<b>B</b>	<b>stop</b>	T
n	<b>B</b>	<b>resume</b>	T

Table 5: CSV format for BeLT system-messages

BeLT only sends events when the application has been started, paused, resumed and stopped.

Mouse events							
Event ID	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	<b>M</b>	<b>M</b>	X Y	T	Event ID	Int	
n	<b>M</b>	<b>U</b>	X Y	T	Event ID	Int	Rectangle
n	<b>M</b>	<b>D</b>	X Y	T	Event ID	Int	Rectangle
n	<b>M</b>	<b>W</b>	Delta	T	Event ID	Int	

Table 6: CSV format for mouse events

**Action** can be:

**M** Mouse move, the next field is where the mouse is now.

**W** Mouse Wheel, the next field will indicate the wheel delta.

**U/D** Mouse press. Mouse Down / Mouse Up, action will say where it happened.

**X Y** coordinates separated by colon(:).

**Delta:** If the event was mouse wheel, value will be the delta value, which says how much it scrolls. Negative value means scroll downwards, while positive value means scroll upwards.

**Flag** Will indicate mouse button on press (1 = left, 2 = middle, 3 = right). The flag will be 4 on mouse wheel and 0 and mouse move.

**Rectangle** The last software rectangle we saw. On down events it will likely be wrong, so you should look at the value given in mouse up events. It still might be wrong. Format is the same as above in the software events. See 5.4.3 for a discussion of weaknesses with our correlation here.

Software events							
Event ID	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	S	FC	Process name	T	Event ID	Element type	Element desc., Element ID, Rectangle
n	S	MO	Process name	T	Event ID	Element type	Element desc., Element ID
n	S	MMS	Process name	T	Event ID	Element type	Element desc., Element ID
n	S	TC	Process name	T	Event ID	Element type	Element desc., Element ID, Extra description
n	S	OCS	Process name	T	Event ID	State	Element desc., Element ID, Rectangle
n	S	EI	Process name	T	Event ID	Element type	Element desc., Element ID, Rectangle
n	S	WO	Process name	T	Event ID	Element type	Element desc., Element ID
n	S	VC	Process name	T	Event ID	Element type	Element desc., Element ID, Flag

Table 7: CSV format for software events

**Action** can be:

**OCS** Object Change State, occurs when the state of an element changes, like checking a checkbox, pressing bold in Word and so on. Can also occur in elements that appear to be buttons.

**FC** Focus Change, occurs when the user shifts focus to a new element, this can be a new window, pressing a button, moving to a textbox and so on. Will indicate

which element receives input on further events.

**EI** Element Invoked, typically pressing a button.

**MO** Menu opened, occurs when the user changes which menu he is looking at, also occurs the first time he starts looking at the menu.

**TC** Text Changed, occurs when the text in an element changes, like an edit box.

**MMS** Menu Mode Started, occurs the first time the user starts looking at the menu.

**WO** Window opened, typically when the user starts a new program or opens a new window.

**VC** Visual Change, occurs when minimizing, maximizing or restoring a window. The extra description will indicate which of the three occurred. Also occurs as restored when the visual state of the window changes.

**Process name** This is the name of the process executable.

**Flag** This is always an integer, it can be:

**state** Says whether the state of the element is pressed or unpressed. 0 = pressed and 1 = unpressed.

**Element type** Says what type of element it is, for a full list see this page <http://msdn.microsoft.com/en-us/library/windows/desktop/ee671198%28v=vs.85%29.aspx> . We subtract 50.000 from each flag, so the button is actually number 0. Negative value indicate that something went wrong.

**Element description** A name that describes what the element is called, this should describe what is the purpose of the element. **UIA\_NamePropertyId**<sup>3</sup> in the documentation for UIA. If the element type is a document or hyperlink, this field will be the URL, or **UIA\_ValueValuePropertyId**<sup>4</sup> in the documentation for UIA.

**Element ID** This is an identifier for the element, it should be unique among all it's siblings. It does not tell you what the purpose of the element is, but it should let you correlate between elements and lets you check if it's the same element you saw before. **UIA\_AutomationIdPropertyId**<sup>5</sup> in the documentation for UIA.

**Extra description** For some events this is an extra description to describe what happened. **UIA\_ValueValuePropertyId** in the documentation for UIA. This field is formatted as a string.

**Rectangle** Describes the area on the screen that the element occupies. Has the following format: < X coordinate of upper-left corner >, < Y coordinate of upper-left corner >< X coordinate of lower-right corner >, < Y coordinate of lower-right corner >, Error is indicated by all negative 1.

**Flag2** As of now, only VC gives another flag, the following values are possible:

<sup>3</sup>[http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017\(v=vs.85\).aspx#UIA\\_NamePropertyId](http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017(v=vs.85).aspx#UIA_NamePropertyId)

<sup>4</sup>[http://msdn.microsoft.com/en-us/library/windows/desktop/ee671200\(v=vs.85\).aspx#UIA\\_ValueValuePropertyId](http://msdn.microsoft.com/en-us/library/windows/desktop/ee671200(v=vs.85).aspx#UIA_ValueValuePropertyId)

<sup>5</sup>[http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017\(v=vs.85\).aspx#UIA\\_AutomationIdPropertyId](http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017(v=vs.85).aspx#UIA_AutomationIdPropertyId)

- 1 Restored
- 2 Maximized
- 3 Minimized
- 4 Unknown – Should never happen

Keyboard events							
Event ID	Event Type	Action	Value	Time	Relation	Flag	Additional fields
n	K	D	value	T	Event ID	flag	
n	K	U	value	T	Event ID	flag	Count if > 1

Table 8: CSV format for key events

**Action** Indicates what type of key event, can be "D" or "U" for key down and key up.

**Value** Roughly what appears on the keyboard, if it is an UTF-8 value. Other keys we can get are system-keys and keys that generate whitespace. See the source code documentation for all possible values.

**Flag** Will indicate which system keys are active. If a bit is turned on it means the following:

1. **bit** Alt is pressed.
2. **bit** CTRL is pressed.
3. **bit** Shift is pressed.
4. **bit** Windows key is pressed
5. **bit** Caps lock is active
6. **bit** Num lock active
7. **bit** Scroll lock active

**Count** Indicates how many key presses was sent. Only sent for KU event, but KD is what is what we mean. The reason for this is that we don't know how many key down events that was sent until we get a key up event. Will be omitted if it is 1.

Hardware events					
Event ID	Event Type	Action	Action specific	values	Time
n	H	KEY	Language	Type	T
n	H	RES	CPU	Memory	T
n	H	SCR_Info	Resolution	ID	T
n	H	SCR	ID		T
n	H	DEV	action		T

Table 9: CSV format for hardware messages

All hardware changes starts with a event ID as before and then the letter "H". Actions can be:

**KEY** Indicates that this is information about the keyboard.

**Language** tells which language and sub-language that is used. 16-bit integer formatted according to this link <http://msdn.microsoft.com/en-us/library/windows/desktop/dd318691%28v=vs.85%29.aspx>.

**Type** is an integer determining what type of keyboard it is. Values can be 1 to 7, formatted after this link: <http://msdn.microsoft.com/en-us/library/windows/desktop/ms724336%28v=vs.85%29.aspx>. (First table under remarks.)

**RES** Indicates that this event shows resources used.

**CPU** average represented by a float value.

**Memory** shows the current memory usage, formatted as integer.

**SCR / SCR\_Info** Indicates that this event is related to the screen. SCR\_Info is used if this is the first time we have seen this screen.

**Resolution** If it is the first time we have seen this screen in this session, it will print out a rectangle, if we have seen it before, it will print out a rectangle representing the resolution.

**ID** is an integer identifying which screen we have changed to. This is unique throughout the rest of the session.

**DEV** indicates that a device has been inserted or removed.

**Action** 1 means that a device has been inserted, 2 means that it has been removed.

## 5.3 Development

### 5.3.1 Documentation

For our project, documentation is an extremely important task, since we are developing a prototype application that later on will be developed even further. But before this happens, our application will be used to collect information from users from a closed set of up to 50 people so our employer can start the task of correlating the collected information and develop an algorithm for validating users. So for this part we have to document our entire system so the system administrator will understand how it works, and what part of the system performs which task.

We decided that the best solution for documenting would be to write documentation for each user group.

For the administrators and developers we created a system manual (Appendix A), it contains the information about how we've set up the system, how we configured the different services and how they work together. In addition to this it describes how to make a new software release.

For the end users we created a manual (Appendix B), it contains all the information about how to use the application – what different buttons do, what does the system tray colors mean etc.

The documentation of the source code needs to be *easy to understand, maintain and scale*. At first we thought of using the XML commenting standard for MS Visual Studio[33], but we found it being too troublesome to work with, also it was no method

for gathering all of the comments afterwards. Instead we opted for the standard that Doxygen[34] uses. When you start a comment with a forward slash and two stars and end it with a star and a forward slash – it will interpret this as a Doxygen comment. See Appendix E to see the entire Doxygen report. Here is an example of a Doxygen comment:

```

1 /**
2  * \brief Initializes all the variables needed to start a session
3  * \author Robin Stenvi
4  * \param[in] bufTmp A string containing the current date and time, with the
   format %Y-%m-%d %H-%M-%S
5  * \remark This function is not enough to start a new session, to start or
   stop a new session, you should
6  * use writeTime with the appropriate parameter.
7  * \returns Returns true on success, false on failure.
8  */
9 bool handleData::startNewSession(std::string bufTmp)

```

Doxygen uses keywords within the comments of the source code. By declaring these keywords it can understand the information and correlate it to the corresponding function/class/member in the reference manual. This makes it easy to write into the source code because all one has to do is write simple one word keywords to add the specified information. The comments are processed and Doxygen automatically generates a formatted reference manual as defined in the configuration file. We configured Doxygen to use Latex, since we also use Latex in our report.

We have configured Doxygen to perform a recursive search of our source code directory, and parse only ".CPP", and ".H" files. Because of this we can add any other directory if we expand our application and still use Doxygen with minimal additional configuration. This also enables any future work to use doxygen without any problems and conflicts. Since our source code for this project is about 10.000 lines of code, a factor to consider is that the documentation must be generated quickly, which Doxygen is able to do.

Doxygen also supports call graphs – call graphs depicts which functions a documented function is calling during its runtime process. Doxygen can create many types of graphs, but the call graph is the most necessary since it gives a very good overview of the flow within the application. To make it easy to understand the source code we have documented function with the minimum of a brief description that states what the function does, we've documented all input values, and return values for the functions and where it is not absolutely elementary what's happening we've implemented a detailed description that describes what the function does. All this is in addition to the name of the author and normal comments, which is left out by Doxygen.

### 5.3.2 Software distribution and Continuous integration

To implement CI we had to create a manual system by creating custom scripts that manage our need for CI. Since we had to manage CI and code analysis from the client side, the need for distribution of new software releases is also managed from the client side with the use of custom scripts.

To implement CI into our development process, without having a build system we could manage our self, was our option to implement CI using a policy driven solution. To do this we set up a small policy, that our code had to be compilable, runnable and reasonably tested before being committed to the repository. To avoid conflicts in the repository we also had to test our code with the latest version in the repository and check

with the others if anyone was working on the same part of the system. If everything was in order the addition of new code was allowed.

The tasks we had to automate during a software distribution, was to create a new version of our software, generate the documentation of our source code and store the current in case later versions should prove to be more stable or there were some other need for the source code.

For the distribution we decided that by using psake[44], we could use the "PSAKE.CMD" application to execute PowerShell scripts with designated psake tasks. With this we could easily perform tasks for compiling the application for different platforms without any problems and on command.

To handle our distribution we created several scripts to automate the process which we have collected in our System Manual, see "Software Distribution" in appendix A.

The main script first compiles our documentation using Doxygen. Then based on its input creates either a x64, x86 or both versions of BeLT with our build script. This stage is initiated by two commandlets, because we had to use psake to help us with the process of building our executables. This stage is then followed by creating the installer for BeLT by running our script for generating the installer using the WiX script, see "Package Management" in the System Manual. Finally we had to gather the current source code before manually updating our update server to accommodate the new version.

To handle our problem of deploying a new version of BeLT inside the repository we had to do the following with our scripts, and manual labour.

- Create new branch
- Generate documentation
- Gather and store current source code
- Generate executables and binaries
- Generate MSI-installer packages
- Upload to distribution server and update deployment web page

Our deployment script performs all of these tasks, except the task of uploading and updating the update server. The deployment script requires that the latest version of the source code is present on the host. The script starts off by creating a new branch for the new version within the directory *"Code/branches/"*.

Then the script continues by generating the reference manual for the system, by running DOXYGEN. Doxygen automatically generates a reference manual for the source code by reading the comments within the source code of the application, as described in section 5.3.2. The resulting PDF reference manual is then moved to the newly created branch.

After the reference manual is created we compile BeLT, by running the corresponding tasks in the "build.ps1" script by using "Psake"[44]. These will then generate a "x64" and/or a "x86" version of BeLT which are moved, one at a time, to the directory *"Code/belt\_main/belt\_installer"*. This is then followed by running the build script for WiX, see section 3.1.4. This creates the MSI-installer packages. The WiX script then moves the MSI-installer packages to a local folder for then later to be manually uploaded to the server for distribution.

This is then followed by cleaning the repository, removing all files not pertinent to the applications source code. Then the source code within *"belt\_main"*, *"dllhook"* and *"belt\_update"*, together with visual studio solution file are archived into a single "RAR" archive with WinRARs<sup>6</sup> command line application. Finally the source code archive is added to the repository.

### 5.3.3 Code Analysis

As explained before in section 3.2.3, the benefits of using static analysis tools are beyond doubt. We have used the built in tool in Microsoft Visual Studio for performing code analysis. This tool goes through the entire projects source code and searches for errors based on the rule set specified.

The build VS12 tool has several sets of rules[45] which we used to detect errors, we normally used the "Mixed Recommended Rules rule set", but towards the end we ran the "All Rules rule set" to detect any remaining errors we've missed otherwise. The "Mixed Recommended Rules rule set"[46] detects common and critical problems within C++ projects, and the "All Rules rule set"[47] contains all the rule sets of both managed and native code. See section 6.1.1 to see how our final test went.

### 5.3.4 Bugtracking

We chose to use Bugzilla[48] for bugtracking because its easy to use and it supports adding enhancements for future work. In this sense Bugzilla is an enhanced bugtracker, because it can keep track of both current bugs and future work. Bugzilla is developed by Mozilla and it's used extensively in their projects, because of this we knew that we where implementing a system that was well tested and reliable.

In our implementation we used our virtual server as a host. We installed and configured Apache, MySQL and a range of required perl-modules. In our installation we followed and used the official Bugzilla tutorials and configurations, except with the email service. We encountered an unknown problem with the sendmail configuration and the result of this was that no email traveled from Bugzilla to its users. We solved this by installing a separate perl module designed specific for Gmail.

The full set-up, configuration and implementation of BeLT is explained in detail in our system manual, appendix A.

## 5.4 Algorithms

### 5.4.1 Mouse compression

Storing every single mouse move event takes up a lot of space on the server and possibly the client, the client application also has to send more data to the server. According to RUI[4], it was 22 KB/m with "extensive mouse movements"<sup>7</sup>. On a full workday, this is about 10MB, just in mouse movements. Since we store a few bytes more on every mouse movement, this number would likely be higher.

We wanted to limit this number while still managing to recreate the original path with good accuracy. The algorithm is based on two variables, the difference in length between two points and the change in degrees between two points<sup>8</sup>.

---

<sup>6</sup><http://www.win-rar.com>

<sup>7</sup>We assume this number is based averages, since they have a separate number for continuous mouse movements.

<sup>8</sup>We only do this on mouse movements, mouse presses and wheel scrolling events are always logged.



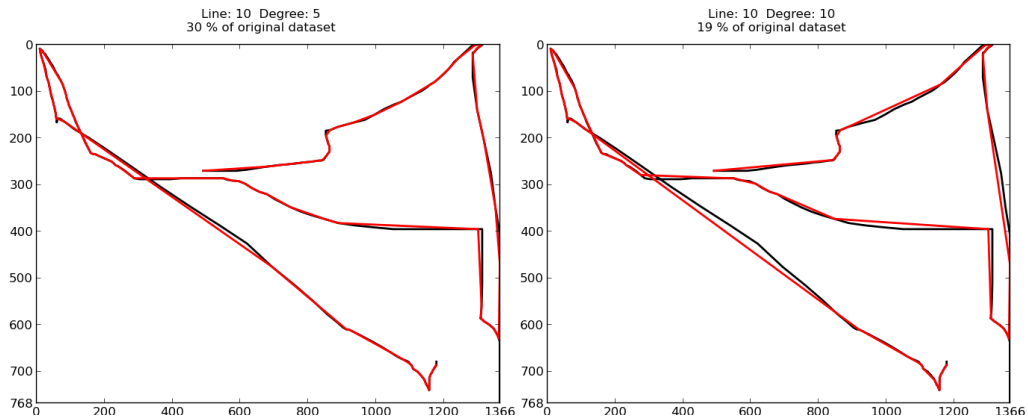


Figure 12: Mouse compression with 30 % of original dataset

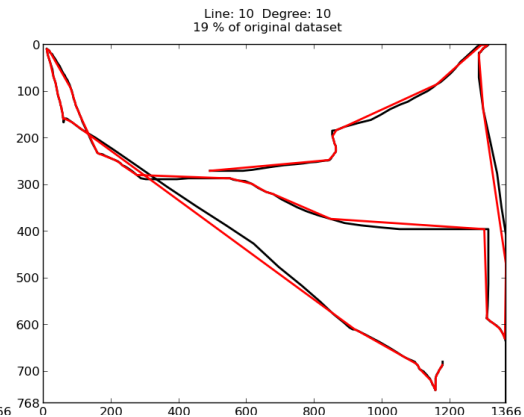


Figure 13: Mouse compression with 19 % of original dataset

To calculate the real distance between two points, we subtract the x and y we logged last with the current x and y values. Both results are squared, and added together<sup>9</sup>. If this distance is higher than some predetermined value, we go on to the next test.

The difference in degrees is measured by taking the current x, y coordinates minus the last x, y coordinates we saw<sup>10</sup>. We then find the arc tangent for this value, multiply it with 180 and divides it by PI to find the current arc tangent degrees. if the difference between this value and the arc tangent from our previous log is above some predetermined value, it have passed the test.

Both of these test have to pass for us to log this event.

Code snippet 5.1 shows our implementation of the compression algorithm. We first check if we have a previously stored point, if not we need to log it.

After limited testing we saw that having a distance change of 10, and a degree change of 5 provided decent result, which is what is used in the final program. We think we can use higher values, but we think it's best to start with modest values, where we know we can get enough data and avoid the risk of not getting enough data.

Code 5.1: Mouse compression algorithm

```

1 REAL_DISTANCE = 10
2 DEGREE_CHANGE = 5
3
4 lastMove.x = -1
5 lastMove.y = -1
6
7 bool procedure difference(b < a)
8     if(b < a) {
9         c = a;  a = b;  b = c;
10    }
11    if( (a + DEGREE_CHANGE) < b)
12        return true;
13    return false;
14
15 bool procedure printThis(POINT now)
16     if lastMove.x == -1 and lastMove.y == -1)
17         return true
18
19     if now.x < 0

```

<sup>9</sup>To get the real distance we would have to take the square root of this again, but it is not necessary for us.

<sup>10</sup>The last x, y coordinate we saw might be different from the last x, y coordinates we logged

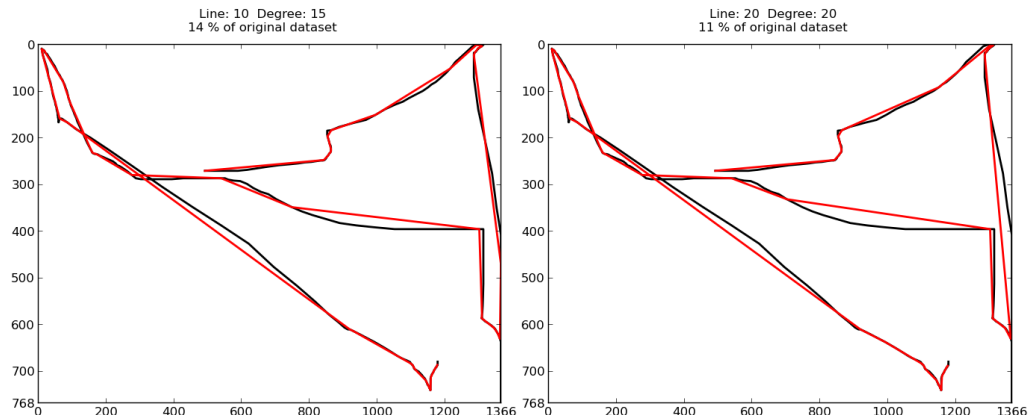


Figure 14: Mouse compression with 14 % of original dataset

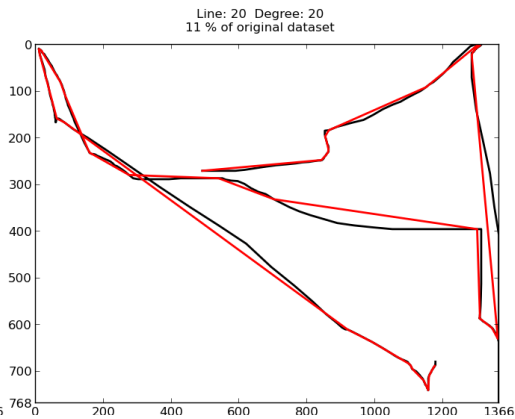


Figure 15: Mouse compression with 11 % of original dataset

```

20     now.x = 0
21     if now.y < 0
22         now.y = 0
23
24     xSquare = square( (now.x - lastWritten.x) )
25     ySquare = square( (now.y - lastWritten.y) )
26
27     if (xSquare + ySquare) > REAL_DISTANCE
28         xAbsDistance = abs(now.x - lastMove->x)
29         yAbsDistance = abs(now.y - lastMove->y)
30
31     newDegrees = arctangent(yAbsDistance, xAbsDistance) * 180 / PI
32
33     if difference(newDegrees, oldDegrees) > DEG_CHANGE
34         oldDegrees = newDegrees
35         return true
36     return false
37
38 if(printThis(now) == true)
39     lastWritten = now
40     Send event to server
41 lastMove = now

```

Figure 12, 13, 14 and 15 shows an example of the mouse compression algorithm. The original path is painted in black, while the path we got from the compressed file is painted in red.

Figure 12 shows the values we used, the other are here to show what kind of differences you can get, depending on how much accuracy you need. The complete dataset contained 440 mouse movements (without any compression). The Python script in appendix D.2 was used to compress each file. Another Python script in appendix D.3 was used to draw the graphs.

These graphs were generated from mouse movements done with a touchpad. This is slightly different from movements done with an external mouse. In our experience, external mouse gives higher accuracy with fewer stored points. This is most likely because movements with a touchpad tends to start and stop a lot, while movements with an external mouse tends to be in a much smoother motion.

Another important thing to mention, is that, there is no guarantee for how effective the mouse compression will be. We did some limited testing, where we tried to do it as realistically as possible, on several systems. The only thing that is guaranteed, is that it

will be as accurate as the boundary values we have set and the mouse events received.

In our experiments, about 35 % of the logs consist of mouse movements, after compression. So if we have a file with 1 MB<sup>11</sup> size. Then 350 KB will be mouse movements, when compression is applied. If mouse compression was not applied, the mouse movements would have a complete size of 1167 KB ( $350/30 \times 100$ ). Here we assume that that we store 30 % of the original data. The new files size would be 1.817 MB ( $(1000 - 350) + 1167$ ).

When running the application in debug mode, it will generate log files for all mouse movements, both without compression and with compression. We did this so it's easy to both test the compression algorithm and check new values. Read more about this in the manual, in appendix A.

#### 5.4.2 Relation between events

When we see an event that is not a hardware event, it belongs to somewhere or is an extension of another event. If we see a mouse down event for example, this belongs to a window, if we see a mouse up event, is is a regular extension of the last mouse down event. The same logic applies to key events. Software events work the other way around, they are caused by a mouse event or a key event, in other words, the relation points to what caused the event.

The relation value is a reference to an event ID in a previous event. It will always point backwards. All the events are placed in one of the following categories.

1. User event tied to window
2. Indirect user event
3. Independent
4. Software event

The first are events that are different depending on which software element they belong to, they consist of the following events.

- Key down
- Mouse down
- Mouse wheel

The second group is events that are direct cause of an event in the first category, it consist of the following elements:

- Key up
- Mouse up

One of these events should always happen when there is a key down or mouse down event.

The third group is just mouse move, since it is independent from other events and does not necessary belong to any window.

The software category is all the UI Automation events. Whenever we get a software event we assume it happened because of a user event.<sup>12</sup>

<sup>11</sup>Whenever we talk about MB and KB in this paragraph we mean  $10^6$  and  $10^3$  bytes, respectively.

<sup>12</sup>This is not always the case, since the operating system does something at regular intervals, plugging in a

For all events in the first category, the relation flag will point to the last active window. For events in the second category, the relation flag will point to its counterpart in the first category. Mouse move will point to zero if this is the first in a series of mouse moves, otherwise it will point to the last mouse move we saw. If a mouse button is held down, the relation flag will point to it, since it will be a dragging action.

The relation flag on software events will always point to the last event that happened in category 1, 2 or 3 (all except software events). This is the flag we are most insecure about, since we have multiple threads and there might be a delay before a new process starts, or a new window is opened.

Table 10 summarizes what we talked about, whenever we receive a key down event, it is tied to a software window. When we receive a key up event it is tied to the last key down event with that specified character. When we receive a mouse down event, it is tied to a software window, mouse up is tied to the last mouse down event. Mouse wheel is tied to the last active software window.

Mouse move can be tied to mouse down, if mouse button is still pressed, or the last mouse move or it is tied to nothing, if user input has happened since last mouse move or mouse down.

On software events, we take an educated guess at what generated it, so the relation flag on software events will point to the last user event, which can be mouse moves, all mouse presses, and all key presses. A guess is all we can do here, there is no sure way, that we know of, to correlate software events with mouse and key events. After talking with employer this was satisfactory, because they would have to do the same thing.

Summary of possible relations values		
Type	Action	Relation
Key	Down	Software
Key	Up	Key Down
Mouse	Wheel	Software
Mouse	Down	Software
Mouse	Up	Mouse Down
Mouse	Move	Mouse Down
Mouse	Move	Mouse Move
Mouse	Move	0
Software	all	Last user event

Table 10: Table of events and their corresponding relationship

### 5.4.3 Area around a mouseclick

Whenever you get a mouseclick, it belongs to somewhere, typically a button, a checkbox or something similar. All these elements are rectangles, and whenever you receive a mouse press, you know that it belongs to one of these rectangles. Some useful properties to know is whether they clicked in the middle of the rectangle, one the left side and so on.

Every time we see a mouseclick, we try to find out which area it belongs to, see table 6. To do this, we look at the last rectangle we saw, which came in the form of a software event. We assume that the mouseclick belongs to that rectangle. This assumption has

USB stick will maybe generate an event and so on. We ignore that here for simplicity.

some uncertainties:

1. The rectangle on mouse down events will usually point to the wrong place, because you first get a mouse down, then an event, then a mouse up event<sup>13</sup>. The rectangle from mouse down is therefore unreliable and the rectangle from mouse up should generally be used, but only if there is a software event between them. You could use the timestamps to increase the likelihood of finding it's correct or not.
2. Normally we receive a mouse event, then the software event, but because we have multiple threads, events may not come in order. If for example, a mouse down event generates a software event; then we would like to see **Mouse down – software – mouse up**, if the software event comes after the mouse up event, both mouse events will reference the wrong rectangle. Events out of order can probably be detected when analysing, because the timestamps will also be out-of-order.
3. Not all mouseclicks generate an event and if an event is generated, and it happened because of the mouseclick, it may have done something with a different part of the screen. If you click an item in the taskbar for example, it will restore a window, but this happens somewhere else on the screen. In these cases the rectangle will be wrong, but in most cases, you will be able to detect it as the mouseclick is not inside the rectangle.

---

<sup>13</sup>This is the general case, sometimes the event occur after mouse up



## 6 Testing and analysis

### 6.1 Tests on client

In addition to using our own test on the application, we also ran our application through "Windows App Certification Kit". We passed this test, with some warnings, you can see the full result in appendix C.

#### 6.1.1 Static analysis

We performed static analysis regularly, so there was no big surprise when analysing the final application. We also performed a manual code review independently within the group to find memory leaks and controlling the errors returned from the static analysis.

We found a couple of warnings during the static analysis test. `GetTicketCount()` was detected as a warning since it will wrap around after 49,7days, but because we already had mitigated this problem we chose to suppress this warning.

In our final version of BeLT we received three warnings when running our static analysis using Visual Studios code analysis tool, with the "Microsoft Native Recommended ruleset". The first warning was a memory leak inside a process handle of `CheckUpdate.cpp`, it came up because we didn't release a process handle before exiting the program. The reason why this wasn't done was because we start a new process which should exit after the parent process exit. This is not a big problem since Windows will release memory when the parent exits the application.

The second and third problem was related to a Mutex error where we manually checked that it was working correctly.

#### 6.1.2 Performance analysis

To measure the performance of BeLT, we did some simple tests. These tests only measure what we consider normal behaviour, meaning that we do them manually while working on some tasks like web browsing or writing text. Since the datasets will be biased with how we work with the computer we will also include what type of events where generated. We still think this gives a better understanding of BeLTs performance than an automated test.

BeLT performance test					
Test:	Avg bytes sent/s	RAM used	Avg. threads	Max threads	avg. CPU
1	1779.67	5-6 MB	14	15	2.5
2	4726.47	5-6 MB	13	16	1.4

Table 11: Summary of our results, when performance testing BeLT client application.

Each test lasted for 15 minutes, was done on a Windows 8 x86 VirtualBox Virtual Machine, with 2GB of physical memory, and 2.4 GHz processor (in total with physical machine, no max limit for how much it can use). The test used BeLT 32-bit version 0.9.0, and BeLT was started right after performance analysis was started. It was not exited until performance test exited.

Table 11 shows a summary of our performance test. The first test we ran, we did 15 minutes of regular writing in Notepad++, while concurrently sending data to the server. The second test was also 15 minutes long and sending data concurrently to the server, this time we did web-browsing in Firefox. We used version 0.9.0 in this test, very minor changes has been done since that.

### First test

The dataset generated consisted of 5111 events, where 974 where mouse movements, 554 where other type of mouse events, 3448 where key events, 129 where software events, 4 hardware events and 2 system messages from BeLT.

On average we sent 1779.67 bytes per second over the network<sup>1</sup>. Most of the time we sent no data at all, but when BeLT has collected 500 events, we send all the data to the server. This can easily be configured to be lower or higher. Network traffic does then increase in bursts, about every two minutes in this test.

Right from the beginning we started at 5MB of private working set, it increased and decreases slightly throughout, but was pretty static.

Throughout the session we never had more than 15 threads running, although several more where created. One is created every time we need to send data to the server, which is for every 500 events, plus 1 for the remaining data when we stop.

The average CPU time was 2.5 %. In the beginning it was as high as 62 %, the reason for this is that we have to register all the logging mechanisms. The amount of CPU we use increases and decreases in bursts, at some points we are as high as 25 %. This is natural, since BeLT only has work to do, when the user types something, or we are sending data to the server.

### Second test

The dataset consisted of 6283 events, where 4960 where mouse movements, 645 where other mouse events, 379 where key events, 293 where software events, 4 hardware events and 2 system messages from BeLT.

This test doesn't differ that much from the first test, we used a bit less CPU on average and we see the same spikes as in the previous test. We see a lot data is sent to the server, but this increase is mostly because the test consisted of web browsing.

### 6.1.3 Time granularity

Time granularity is very important for behavioral analysis. When looking at timestamps, we consider two factors:

1. The timestamp is reliable – at some point in time when the algorithm is created – there is no real way of detecting anomalies, so you need to be able to trust the timestamps that are generated.
2. High resolution on the timestamp – with high resolution we also mean that the counter must be updated often. The timestamps we get in our messages has a resolution of 1 ms, but the counter is only updated every 16 ms.

Every time we get a mouse event or keyboard event, we also receive a timestamp for when the event happened. The timestamp we receive, is what is called "Windows time" and is the same time that is returned by the following function[49, 50]:

---

<sup>1</sup>This might include some data that was not generated by BeLT, but not much



```
1 DWORD WINAPI GetTickCount(void);
```

This function gets the time in millisecond, but the resolution is limited to the system timer, which according to Microsoft is in the range of 10 to 16 milliseconds. This roughly coincides with our tests. Our test shows that when using a program to constantly generate 2000 key events without pause, we get 9 different timestamps, where the smallest difference in timestamp is 15 milliseconds and the biggest is 16. The biggest difference is not that important, because it can be greatly influenced by changes in the processor. The smallest difference and the number of unique timestamps says a lot about the granularity.

To get more fine grained granularity, we can collect our own timestamp, to get an accurate timestamp, according to Microsoft[51], we can use the following function:

```
1 BOOL WINAPI QueryPerformanceCounter(
2     _Out_ LARGE_INTEGER *lpPerformanceCount
3 );
```

The granularity here is also dependent on the system it is running on. We can retrieve that number by using the following function[52]:

```
1 BOOL WINAPI QueryPerformanceFrequency(
2     _Out_ LARGE_INTEGER *lpFrequency
3 );
```

This cannot change while the system is running, so if we run this at startup, we can interpret the remaining numbers we get from the previous function. On our system, which was Windows 8 running in VirtualBox, the frequency was 2.329.681 counts per second. This means that the counter will be updated roughly twice every microsecond. This also shows when using this timer against the same test as before. We get 2000 unique timestamps, so every timestamp has a new value.

To test how accurate timestamps, we run our own simple test. It consisted of generating 2000 key strokes and sending that to our own logging application that only stored each timestamp.

Code 6.1 shows how we generated keystrokes, the important part is that we generated them as fast as possible, without any pause.

Code 6.1: C++/pseudocode for time granularity testing

```
1 CreateProcess(belt_main.exe)
2
3 for(int i = 0; i < 1000; i++) {
4     keybd_event(0x41, 0, 0, 0);
5     keybd_event(0x41, 0, KEYEVENTF_KEYUP, 0);
6 }
```

Code 6.2 shows how we measured time granularity using the timestamp we get from each key event. Code 6.3 shows how we retrieved timestamps separately.

Code 6.2: Section 1 of logging program for time granularity testing

```
1 void writeRegular(KBDLLHOOKSTRUCT hook) {
2     file << hook.time << "\n";
3 }
```

Code 6.3: Section 2 of logging program for time granularity testing

```
1 void writePerformance() {
2     QueryPerformanceCounter(&count);
3     file << count.QuadPart << "\n";
4 }
```

Below you can see how each test went. In the first test, using the timestamp we get from the key event, we got 10 unique timestamps where the smallest difference is 15 milliseconds. The second test is much more accurate, each timestamp is unique and the smallest difference is 2. The biggest difference is not that important in either cases as it can be attributed to scheduling or many other things that happen at the computer simultaneously.

The first test lasted for about 141 milliseconds, while the second test lasted for about 125 milliseconds.

```
./print_stat.py keylogRegular.txt
Read 2000 timestamps
10 unique timestamps
Average difference is 15.6666666667
Smallest difference is 15
Biggest difference is 16

./print_stat.py keylogPerformance.txt
Read 2000 timestamps
2000 unique timestamps
Average difference is 146.776888444
Smallest difference is 2
Biggest difference is 4947
```

The source code for "print\_stat.py" is in the appendix D.1.

This test shows some important differences between the timestamp we receive from Windows and the timestamps we can generate ourselves. One obviously has higher resolution than the other. It seems like the counter on this system is updated every 15 ms, because: The test lasted for 141 ms, which means that the counter should have been updated  $141/15 = 9.4$  times, during that period, which coincides well with our result of 10 unique timestamps.

If the counter is updated every 15 ms, we should be able to generate a key event every 15 ms and receive 2000 unique timestamps, the new test has the following code:

Code 6.4: C++/pseudocode for generating key events on second test

```
1 CreateProcess(belt_main.exe)
2
3 timeBeginPeriod(1);
4 for(int i = 0; i < MAX_ONE; i++) {
5     keybd_event(0x41, 0, 0, 0);
6     Sleep(15);
7     keybd_event(0x41, 0, KEYEVENTF_KEYUP, 0);
8     Sleep(15);
9 }
10 timeEndPeriod(1);
```

The function to *timeBeginPeriod*[53] is important to notice as it changes the resolution of the sleep function from 10-16 ms to 1 ms. Which is the reason we are able to have high precision intervals.

The results are as expected:

```
./print_stat.py keylogInter.txt
Read 2000 timestamps
```

```

2000 unique timestamps
Average difference is 16.6333166583
Smallest difference is 15
Biggest difference is 281

```

This test lasted for 33 seconds and 596 milliseconds, which means we, on average, spent 16.798 ms on each key event. Since this gave us 2000 different timestamps, we can conclude that timestamps appear to be reliable and reasonably accurate.

We didn't run the second test on our other method of generating timestamps. The reason for this is that it is much harder to test – because we know that we will get all unique timestamps and that a difference can be influenced by a great number of things, like scheduling. Therefore we don't think that test would provide any analysable results.

One problem with the last method of generating timestamps that we haven't discussed yet is that we have to generate those timestamps ourselves. To make it accurate we have to generate it as fast as possible after an event happened. We are hooking into the messages, these messages are sent in a FIFO (First in first out) queue. The accuracy of our timestamps is both dependent on when we receive the message and how fast we are able to process the message. If this delay was the same each time there would be no problem, but we can't guarantee that[54].

We might be able to mitigate this problem by keeping two timestamps, one performance counter and one "Windows time". We know that the "Windows time" is accurate up to 16 ms – so if the timestamp from the performance counter shows a later time than this we know that is not accurate. This has not been done in our application.

Another small problem is complexity. The resolution of performance counters are system dependent (which can vary), this makes the analysis harder. But this is not a big problem, since you can reduce the resolution of the timer, to 1 ms.

## 6.2 Performance optimization on server

The client application generates a large amount of events. This happens more or less continuously. Therefore we need to make sure the server can handle this amount of data, and the client must be able to process it without it causing a delay.

We use Syslog-NG as the server component, so one part of dealing with the performance on the server is to configure Syslog-NG to good performance, while minimizing resources it uses and the risk of losing data. Some inspiration has been gathered from [55].

### 6.2.1 Configuration

Some interesting configurations to consider are the following:

**flush\_lines():** Here we can set the number of lines that come in before the server flushes the output to file. This offers better performance, but increases latency and might cause data loss if the server crashes.

**flush\_timeout():** Here we can guarantee that data is flushed in regular intervals, regardless of what our previous value is. This causes a small hit to performance, while it minimizes latency.

**log\_fetch\_limit():** This sets how many lines Syslog-NG will fetch from a loop. If we read more lines at a time we gain performance.

## 6.2.2 Performance test

This test is only meant as a small scale test before the real simulation in section 6.3. For this test to be more effective we would need to run it for longer time with several more options. But it does give us a decent baseline as it stands now. Now we only test one type of simple output, so we don't test any databases, any filtering or any of the options we will use in the future. Code 6.5 shows the relevant parts of our current syslog-ng.conf file.

Before starting the research project, a larger test should be run. That test should be on the hardware that will be used in the project and use the final data format. That way we get as close to the reality as possible.

Code 6.5: Relevant part of syslog-ng.conf file

```

1 options {
2     chain_hostnames(off);
3
4     # It HAS to be <= log-fetch-limit / max-connections
5     flush_lines(10);
6     use_dns(no);
7     use_fqdn(no);
8     owner("root"); group("belt"); perm(0640);
9     stats_freq(0);
10    bad_hostname("^gconfd$"); create_dirs(yes); dir_perm(0750);
11    keep_hostname(yes); # Keep unique ID instead of IP
12    frac_digits(4);
13 };
14 #####
15 # Logs from TLS, which is what we use by default
16 source s_tls_remote_no_auth {
17     tcp(
18         ip(0.0.0.0) # All IP addresses are accepted
19         port(1999) # Uses port 1999, 19155 from the outside
20
21         # Messages are structured as RFC5424
22         flags(syslog-protocol)
23         keep-timestamp(yes) # Keep the original timestamps
24         tls(
25             key_file("/home/ca/private/belt.key.pem")
26             cert_file("/home/ca/certs/01.pem")
27             peer-verify(optional-untrusted)
28         )
29         max-connections(1000) # Total number of users
30         log-iv-size(100000) # Is divided by max-connections
31
32         # Max messages fetched from a single loop
33         log-fetch-limit(100)
34     );
35 };
36 #####
37 # Simple destination for performance testing
38 destination d_performance_test {
39     file("/var/log/TEST/performance_test.log");
40 };
41 #####
42 # Log description for TLS communication
43 log {
44     source(s_tls_remote_no_auth);
45     destination(d_performance_test);
46     flags(flow-control); # Never throw away message
47 };

```

To test the server, we will use a tool called "loggen" that is shipped with syslog-ng. This can be used to generate a series of log messages, we will use the following command

in all our experiments:

```
1 /usr/bin/loggen -iU -r 10000000 -P -R test_file.log -l -d -I 180 127.0.0.1
  1999
```

Below is a description of what each parameter does.

- iU – send packets using IP and TLS.
- r – number of packets per second, here it is 10 million.
- P – format packets according to the syslog protocol.
- R – use messages from file.
- l – loop over the file as many times as it needs.
- d – print each line in the file regardless of how it is formatted. This makes it easy for use to use our own file that we know is compatible with our parsing. This is especially useful if you also want to test this with your own message parsing, which we don't do here.
- I – interval, which here is 180 seconds or 3 minutes.

We want to use our own message file to get the most accurate results. The main difference from this and BeLT, is that "loggen" will send messages as fast as possible, which is not the case when BeLT is distributed in the real world. It does however give us an overview of how many messages it can process per second.

Table 12 shows how the test went. Each test only changes one value. When we move on to a new setting we use the most successful value from the previous setting.

Output from performance test		
Setting	Msg/sec	Interpretation
Base test	26 817	Base test with default configuration
log-fetch-limit(1)	16 1701	Decreasing the log fetch limit from 10 (default) also decreases the average rate
log-fetch-limit(100)	27 411	Increasing it to 100, increases the rate slightly
log-fetch-limit(200)	22 093	Increasing it further, decreases the rate
flush-lines(10)	30 324	Increasing the flush lines from 1 (default) to 10, increases the rate significantly
flush-lines(100)	21 504	Increasing it to much, will decrease the rate

Table 12: Output from performance test

This was a simple performance test to see how different settings would increase and decrease the performance and also see how many events Syslog-NG could handle with very basic settings. A more rigorous test would run for longer than 3 minutes, would change several more settings and use the final format.

### 6.3 Server testing

As part of our testing scheme we had to perform a stress test on our server for several reasons. First of all we had to test the different ways of storing the large amount of data we were going to send. Secondly we had to figure out if our Syslog-NG server would

be able to handle a hundred users simultaneously without affecting the performance of the server. Third we had to ascertain how much storage space would be required by the format. Thereby figuring out what storage format would be the most effective for the system to handle.

We wanted to test five separate methods of storing our data. First we wanted to perform the baseline test which were to try to only output the received data and nothing more. Basically just printing the raw data. Secondly we wanted to test how a CSV formatted file, and a XML formatted file would perform. The last two methods we wanted to test was the unindexed and indexed database storage systems. All in all we wanted to figure out which format was best suited for this data.

To do this we needed to measure quantifiable data on the server to depict its performance, the data we gathered was CPU, RAM, and I/O statistics. To capture this information we installed SAR[56], a tool to monitor and log the system performance at a fixed interval. With this interval set to 1/sec we would get a dataset consisting of about 3600 data points for each hour long test. The resulting dataset would with this information contain more than enough data for us to calculate the load the server was exposed to.

Since we had five different storage formats to test we wanted to run the test automatically, which we solved by creating a bash script on the server. On the client side we created a software application that simulated a fixed amount of users sending data to the server, which we set to five. This is because simulating more users would reduce the performance of the computer and thus render the test invalid. We chose to emulate five user because this was a number that wouldn't cause any performance issues and the networking capability wouldn't be affected either.

### 6.3.1 Methodology

Our testing methodology was simple. We set up a range of host computers running our simulation application. Inside the application each simulated user uses data from a fixed file to generate and send data to the server concurrent with each other until the test is signalled to be finished. This is done while the SAR program runs in the background, logging the servers performance statistics once per second.

The drawback of our testing methodology is that it doesn't take into account for different amounts of users above the threshold we tested. This is discussed more in the results and conclusion, section 6.3.5. This is because we performed the test with the maximal amount of users we could simulate within that environment.

Prior to starting the test on the server, we had to manually initiate the simulation application on the host computers. Then when all hosts had started listening to our test controller, a website with a predefined content told the host whether the test was starting. When the server script initiated the test by changing the content of the website, the hosts would establish a connection and start sending data by iterating over the same log file for one hour. Which was then followed by the server having six minutes to settle down. This would ensure that all users had finished and events was parsed before setting up and initiating the next test, doing the same until all the test had finished.

The bash script that controlled the server test is shown in appendix D.6.

### 6.3.2 Test flow

To test the performance limit of our server we wanted to test with at a hundred users sending data at the same time. Since we could not simulate a hundred users on our own

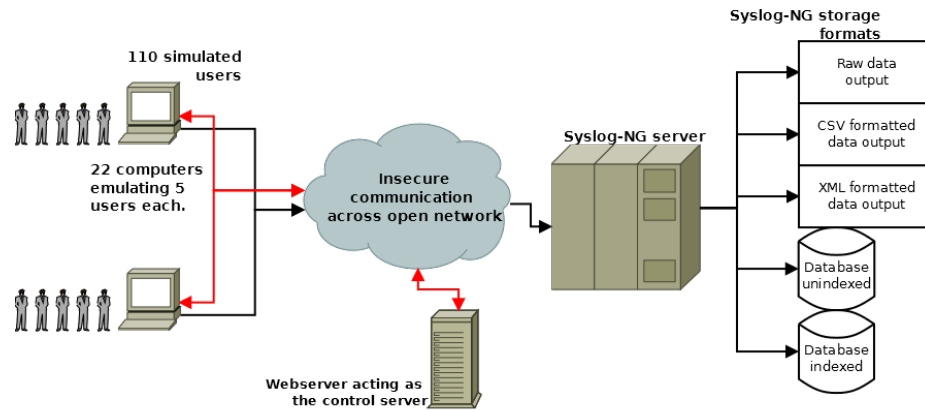


Figure 16: Depiction of our server test setup

laptops, our alternative was to acquire a whole computer lab at GUC during a weekend in February. The lab consisted of 32 desktop computers(hosts), where only 25 hosts were usable. Our plan was to use all the hosts and simulate 125 concurrent users, but because of an error we were only able to use the data collected from 22 hosts during the long term test. This was because three of the hosts went to sleep and didn't retrieve the stop message, when we awoke them during our systems check between the first and second test. This caused them to continue sending data to the server when the test was over. We then had to terminate the hosts before the second test started. As a result we continued our tests with 22 hosts before running the first test once more after the others was finished.

In figure 16, we've illustrated the setup of our test. First we had our Syslog-NG server running as the recipient of all communication from our 22 hosts, simulating five "users" each. Upon receiving the data, Syslog-NG would perform the currently, configured task in the config file and store it in the corresponding file format. The file formats we wanted to test was raw output, CSV formatted output, XML formatted output and both indexed and unindexed MYISAM database.

The raw output, CSV and XML formatted output was all handled by the Syslog-NG servers configuration files. The database storage functionality was on the other hand done using a separate script. The reason, this was not done with Syslog-NG is that it didn't provide the necessary functionality. Instead we used a pipe within the Syslog-NG config file. This wrote the data to a FIFO list in "var/tmp" where the data was read by a bash script looping and reading from the FIFO file. For each line the bash script reads, it sends the text as standard input to the MySQL service with the following syntax, "*mysql -u syslog -password=\*\*\*\*\* < \$ line*". Since Syslog-NG automatically formats the data it receives according to its configuration, we had a configuration which formed the data into a MySQL procedure call. Therefore by sending the read text as standard input to the MySQL service we called our custom procedure that inserted the data into the appropriate fields in the database.

We've illustrated the flow of our testing in figure 17. Yellow represent our test control script running on the server, managing the update of config files and content on the webservice. The blue webservice is a normal Apache service that contains a file which is readable from the Internet. This files content is changed to show that the test has either

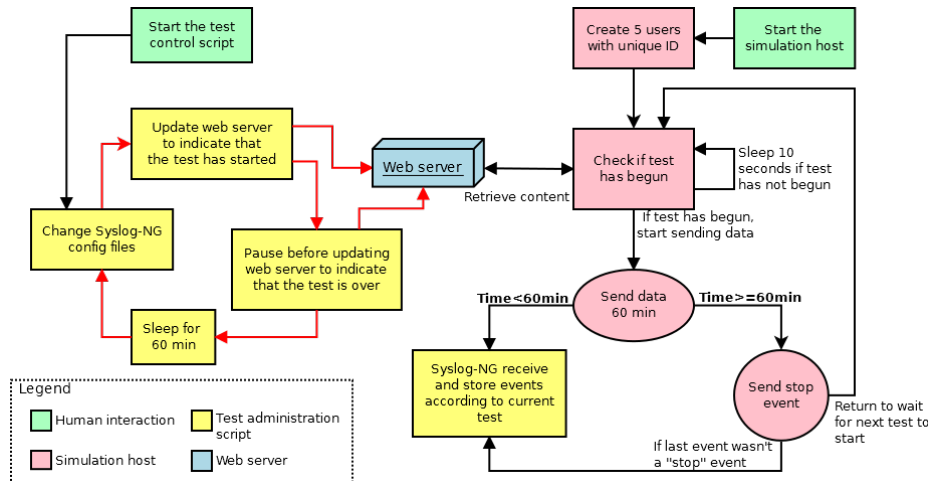


Figure 17: Flow chart for how the testing was performed

started or is stopped.

The green fields represent human interaction which is us, when we started the server test by initiating the server script. It also depicts that we started the 22 hosts running the simulation application prior to starting the test on the server.

The pink fields illustrates the simulation application running on the test computers. Which is responsible for creating users and send information to the server during the different test.

### Simulation application

The first stage in running our test was to start each and every computer host running our simulation application. This had to be done prior to starting the test control script because it would cause a host to not connect, and therefore never send any data. Which would cause inconsistencies in our testing.

When the simulation application is started it first creates five "users" with each of them their own unique ID. Then the simulation application will check if the test has begun by connecting to the webserver and retrieve the content of the webpage. If the content indicates that the test has started it will proceed and start sending data for 60 minutes. This task will run for an hour and send events defined by a previously generated log file. As long as the timer is below 60 minutes the simulation application will continue to transmit data to the server, generating new sessions for each complete iteration of the predefined log file.

Then when the timer reaches 60 minutes it will check whether the last event sent to the server was a "stop" event. If not, it will send an additional packet to the server containing a "stop" event before returning to check if the next test has been started.

As long as the content retrieved from the web server indicates that the test has not started, the simulation application will pause for 10 seconds before once again returning to check whether the next test has begun. Then when the retrieved content indicates that the next test has started the user will proceed doing the same all over again.



### Test control script

After all the simulation applications were started on the hosts we started the test control script. The first thing the test control script does is to change the configuration files for Syslog-NG to ensure that the correct fileformat is generated. After replacing the existing configuration file it restarts the syslog service . Then the script starts SAR in the background to log the system performance.

Following this the server replaces the content of the file on the webserver content that indicates the test has started. The test control script then sleeps for a short period, to ensure that everybody has connected and has access to the server. When the test script awakens once more it changes the content of the webpage to indicate that the test has ended.

After the test script has set the content of the webpage to indicate test stop, it goes to sleep for a 60 minutes before awakening and sleep one final time, to make sure that that all events has been properly parsed and stored. After this the test control script will return to replacing the current Syslog-NG config file with a new one configured for a different file format

### 6.3.3 Running the tests

Prior to the first test on Friday the 22nd of February we wanted to test our system to ensure that it ran smoothly, without any problems. The tests we ran was five minutes long for each storage format to ensure it worked properly. This way we improved the test scripts, simulation application and Syslog-NG before we ran the test on Friday. Since these tests were done using a single computer, simulating five users for a few minutes, we were only able to test the basic functionality and flow of our testing methodology.

#### First test

On Friday we started out with trying to set up our testing environment at the computer lab, where we had a couple of problems. First we were missing DLLs when just copying the executable onto the hosts. To fix this we had to install a separate program on the hosts, but the terms set for our use of the computer lab stated we couldn't install software on the computers. So after a few trials we were able to create a software that would function by copying it onto the host without any need for external software, that we didn't include ourselves.

Then after getting the test environment up and running we started the hosts to run a short test where we ran each storage format for five minute before continuing with the next format. Right after starting the trial test we lost connection with all except for ten hosts. We then restarted the five minutes test with the ten remaining hosts, which worked without a problem.

When the five minutes test was finished running we engaged the hour long test, where each storage format is running for one hour before sleeping for a little while and continuing with the next format.

As a result of the short test we ran the hour long test with the ten remaining hosts that didn't fail during the first trial. Along with the Syslog-NG and the scripts we also ran SAR in the background. SAR logged the performance statistics of the server every second and stored it so we could later on evaluate the results of each storage format and see how they affected the server.

The results which we analyzed on Saturday gave us the insight that using ten hosts

to emulate fifty users was a big underestimation of what our server could handle. The results the test yielded is not published because the system performance was not affected, thus not yielding any valuable data.

We decided, based on this to run another test on Sunday with all available computers in the computer lab, after configuring the Syslog-NG server to accept more than ten connections.

### **Final test**

After our failed tests on Friday we mitigated the error in our Syslog-NG settings to allow for more connections at the same time. Which meant we could use all of the functioning computers in the computer lab.

When we had set up our test environment with the 25 functioning hosts we started the test with running each storage format for an hour before switching. This ran fine for the first storage format, but when pausing before starting the next format we discovered that three hosts had not stopped sending data to the server. We then stopped these host from sending data before the next test started.

There was also a fourth host that was running an older version of our simulation software, but it did not cause any problems for our testing purposes, nor for our data collection. All in all we had to shut down three hosts during our first test, which left us with 22 hosts to run the test with. The results from the remaining 22 hosts are what we have based our results on. The tests we ran was in the end, despite some minor problems, a complete success. There was however a problem that occurred with SAR because our server were running as a VM on a VM server. The VM server's scheduling algorithm would then delegate its resources between the currently running VMs which caused us to loose a significant amount of from SAR, but we were still able to generate our graphs and calculate the averages of our servers performance statistics.

The biggest problem however was the test we ran on our indexed database implementation, where we lost 753 packets from Syslog-NG. The error was caused by us when we prematurely ended the test to restart our first test, as explained in detail later.

## **6.3.4 Flaws in our test**

### **First test**

On Friday we initiated a small scale test with 20 hosts running our simulation software. Once we had started the simulation software on the hosts everything seemed okay and we initiated the server. When we started the server vi instantly lost connection on half our hosts and were forced to abort the test session.

After a while of error searching we figured out that our server could have a limitation on the number of simultaneous connections of 10 hosts, and since we were unable to find the error in our configuration we decided to run the long test using only 10 hosts. This time the test was run successfully and all the data was collected successfully.

Because of this we decided to take a look at the data, fix the server and decide whether we needed to run a bigger test. So on Saturday we fixed the error in our Syslog-NG configuration that hindered more than ten connections at the same time. This opened up for more connections so we could use all of our hosts. After lifting the connection limit for TLS-connections on our server we analyzed the data, which showed that the resource usage on the server was too low to draw any significant conclusions.

## Final test

During our long term test on Sunday, where we let the hosts running the simulation application, send data to the server for a full hour for each of the storage formats. During this test we started out with 25 hosts, but when the the first storage format had finished testing three hosts would not stop sending data to the server. We caught this error prior to starting the test of the second storage format and shut the hosts down. Because of this error the hour long test was no longer valid, but all later tests with 22 hosts was successful and error free. To mitigate the error we ran the test of the first storage format once more with 22 hosts, after all other formats had finished, thus getting a complete set of valid and correct data without errors.

What had happened was that we had forgotten to configure these three hosts to never enter hibernation/sleep mode and never to shut off the screen. After a while they went to sleep, and when our first test was over we went to check on the hosts. When we awoke the three hosts they continued to send data even though the test was over. This render our test for the raw storage format invalid. We then quickly shut the hosts down, before the second test started, to avoid corrupting our second test. We then ran the first test again after all other tests was finished to get a complete set of valid tests.

Our problem with a fourth computer was that we were using a an older version of our simulation application because of a glitch when replacing the previous version. Even though it didn't end up causing any problems we had to search through all of our records and databases to look for whether an error had occurred. After a systematic search of our databases and files we didn't find any errors at all that had compromised our results and data sets, except for a man made error explained below.

Another error during our testing was that SAR lost a significant amount of data, see table 14, but because the data lost was spread out through the test we could still use it to generate averages. This happened at regular intervals because the underlying server would schedule its resources to another VM and therefore cause a gap in the data sets when we hav no access. Even though we lost about 20% of the data for each test, we were still able to retrieve the data we needed because it wasn't a consecutive loss of data. It was instead small losses throughout the test. The losses that occurred is best explained by looking at figure 19, of our CSV test. It shows that the amount our CPU spent being idle drops to zero at regular intervals. The effect of this is that when the server schedules resources our way again, we'll have a queue of events waiting to be parsed. This will cause a drop in our idle time until we have parsed the queue.

As we stated previously the only real error that occurred was when we restarted our hour long test for the raw data. What happened is that the test finished for our indexed database and we shut down the server so we could restart the first test. When doing this we were to quick and shut down the server before the remaining events in the queue was inserted into the database. We found this out when we later on were gathering data for our summary table, table 13.

After finding the sessions that had where missing events, we started investigating what could have caused this. Our theory was based on that we had a total of 753 missing packets and there was 75 sessions missing their last "stop" event in the database. Based on this we believed we had terminated our test prematurely and caused 75 sessions loose 10 events each, since this would correspond to 750 events out of the 753 missing. This was further investigated and confirmed by the following SQL-queries:

## Code 6.6: Select all session without a "stop" event

```

1 SELECT event.SID, event.UID, event.MSG, T2.MSG
2   FROM event LEFT JOIN (
3     SELECT SID,UID, MSG FROM event WHERE MSG="stop"
4   ) T2 ON event.SID=T2.SID AND event.UID=T2.UID
5   WHERE event.MSG="start" AND T2.MSG IS NULL;

```

Based on this query we found out we had lost at least the last package for 75 sessions since it returned 75 rows with a session that did not have a corresponding "stop" event.

## Code 6.7: Select all distinct user ids from sessions without a stop event

```

1 SELECT DISTINCT event.UID
2   FROM event LEFT JOIN (
3     SELECT SID, UID, MSG FROM event
4     WHERE MSG="stop"
5   ) T2 ON event.SID=T2.SID AND event.UID=T2.UID
6   WHERE event.MSG="start" AND T2.MSG IS NULL;

```

Based on this query we discovered that all sessions belonged to a unique user, since it returned returned 75 rows where the UID is distinct and "stop" event is missing.

## Code 6.8: Check if anyone are missing multiple sessions

```

1 SELECT COUNT(event.SID) as SIDX, event.UID as UIDX, event.MSG, T2.MSG
2   FROM event LEFT JOIN (
3     SELECT SID, UID, MSG FROM event WHERE MSG="stop"
4   ) T2 ON event.SID=T2.SID AND event.UID=T2.UID
5   WHERE event.MSG="start" AND T2.MSG IS NULL
6   GROUP BY UIDX HAVING SIDX > 1;

```

The query shows that no user have multiple sessions that are incomplete, since the query returned zero rows.

## Code 6.9: Check if the session without "stop" event is the last session

```

1 SELECT event.SID as SIDX, event.UID as UIDX, event.MSG, T2.MSG
2   FROM event LEFT JOIN (
3     SELECT SID, UID, MSG FROM event WHERE MSG="stop"
4   ) T2 ON event.SID=T2.SID AND event.UID=T2.UID
5   WHERE event.MSG="start" AND T2.MSG IS NULL
6   HAVING SIDX = ( SELECT MAX(SID) FROM event WHERE event.UID=UIDX );

```

Based on the query we found out that each of the 75 incomplete sessions is the last session for its user.

Taking all of this into considerations we are certain that the loss of data was caused by a prematurely shutdown of the test, and not any configuration errors or system failures. All in all our error caused the server to loose 753 events from 75 unique users from their last session, as proved by the SQL queries above..

### 6.3.5 Results and conclusion

As previous stated we reached our max resource usage when using the indexed MYISAM database. The reason for this is somewhat due to the fact we piped the formatted data to a FIFO-list which was then read by a bash script that opened and closed a connection to the MySQL server for each line it read. We would most likely have received better results if we had used an application that would keep an open connection to the MySQL server, but database storage would still require the most resources especially when using an indexed database. For a full summary of our results see table 13. For an explanation of each storage format, how they were configured and implemented is explained in section 6.3.6.

Because we used a virtual server, we got some errors in our statistical data which we wouldn't have received if we had used a stand-alone server, which would also have resulted in cleaner graphs when depicting resource use. The best file format to store in would, based solely on our statistical results, be CSV. This is because there was very little difference between XML, raw and CSV in the performance statistics. We didn't lose any events and the storage size of CSV is by far the lowest.

With the future in mind we can surely state that the data set will grow far beyond a manageable size with a manual file directory it will be need to implement database storage. This is more easily done with the XML file format and is by far the best way to go when it is going to be implemented. This is because the information in the XML file is more easily identifiable and easier to work with since there are many APIs for any language that can parse XML files. The drawback with XML is that it requires almost three times the size of a CSV formatted file, but this is only a small problem with today's possibilities for storage space.

The raw format is not usable because there is no structure of the events, just pure output from Syslog-NG without any structuring or processing, and is only used to provide a baseline for the resource usage. The database storage is not usable because it requires too much resources and would not scale well, but it had the advantage of putting information straight into a database which then were ready straight away. Though, even with a service running an open connection to the database, would not be the best solution since the processing is much heavier than outputting the processed Syslog-NG formats to a file.

Instead, importing data to a database for storage should be done by running a scheduled task during hours of inactivity on the system, i.e. during the evening, night or weekends, see section "Data Export" and "Data Import" in section 4.3.2.

Data summary								
Test	Sent	Events/s	Lost	CPU avg	CPU idle	Size B/event	IO s/sec	request-
Raw	2 593 615	704.7/s	0	4.8	62.30	116.9	8.19	
CSV	2 273 460	617.7/s	0	4.8	62.39	55.4	8.44	
XML	2 400 220	652.2/s	0	5.1	60.46	150.0	8.59	
Unindexed DB	2 289 575	662.1/s	0	43.3	41.43	126.4	4.28	
Indexed DB	1 645 475	447,1/s	753	48.2	4.29	265.8	37.34	

Table 13: Summary of our test results.

Table 13 shows the summary of our test. The "Sent" column represent how many events were sent from the simulation application on our 22 hosts. "Events/s" is how many events was sent per second during our a bit over one our long test. "Lost" states how many events were lost during the test. "CPU avg." represents the averaged CPU use and "CPU idle" represents the average amount the CPU spent in user mode and idle/waiting mode, respectively. The "Size B/event" column represent how much storage space a single event requires on the server. The "I/O requests/sec" represents how many input and output requests was made per second on the server.

SAR loss statistics				
Test	Capture time	Data points	Amount lost	Percentage lost
Raw	05:50:00 – 06:51:26	3680	679	18
CSV	01:34:22 – 02:35:42	3680	800	22
XML	02:35:44 – 03:37:04	3680	823	22
Unindexed database	03:37:06 – 04:38:26	3680	824	22
Indexed database	04:38:59 – 05:38:56	3680	795	22

Table 14: Number of losses within each hour long test

Though our tests was successful it only shows one real test with 110 users. This makes us unable to depict how the resource usage will behave when i.e we double the amount of concurrent users. Because of this we can't say for sure if the resource usage scales linearly or exponentially when adding more users, but with 110 users generating an average CPU user mode load of 4-5% along with the CPU being idle 60% of the time are we confident that the server could handle at least 400 users for the XML, CSV and raw output formats. In addition to average I/O wait being 40% for XML we know we have a lot to go on when it comes to available resources.

The virtual machine which ran our Ubuntu 12.04 server wasn't the most powerful of servers. It ran on a single virtual CPU with 1GB of RAM and 10GB of storage space. The processor on which it ran was an Intel(R) Xeon(R) CPU E7-4830 @ 2.13GHz. All in all this isn't a bad setup for running our development project, but in a production environment we would want to implement a system with multiple log servers to allow for load balancing. For our case we would have implemented a lightweight load balancing system which granted one user a fixed server to log data to, then users would be distributed equally among the servers to avoid congestion on a single server.

Since we where able to handle 110 users with great ease during our testing we don't see any problems with having our development server to be used for the initial data collection to come.

### 6.3.6 Explanation of test

#### Raw output

The raw data output means that Syslog-NG only output a long string to the file corresponding to the sending users session id. With all the data separated, by a comma, as shown in listing D.7. This is test was run to use as a baseline value since this required the least amount of parsing of received data.

Figure 18 describes how much percent the processor spent being idle during the test. The results are conclusive, and shows us that we still had plenty to go on when it comes to available CPU resources. This means we could run the test with at least as many users as stated in the previous section 6.3.5 before any severe performance issues would appear.

#### CSV-formatted output

Figure 19 describes how much of the processor spent being idle during the test. The results are conclusive and depicts that we still had a lot to go on when it came to available CPU resources. This means we could have run this method with a at least the amount stated in the conclusion, section 6.3.5. This mostly because the file format doesn't require

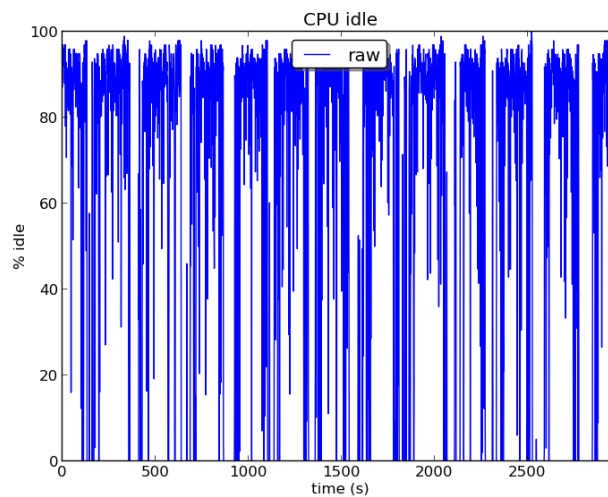


Figure 18: Percentage of time spent idle when using raw mode

much processing, even though it is much larger and more complex than the exportation for raw output.

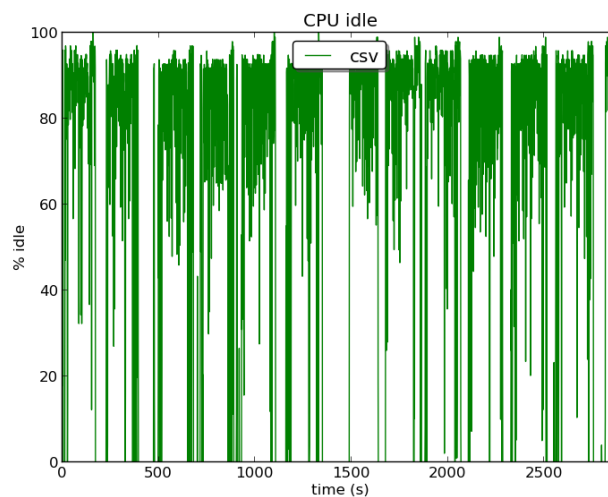


Figure 19: Percentage of time spent idle when using CSV

The output to create CSV files means we had to create the files according to a specific format, described was developed in collaboration with our employer. This file format is much lighter to store than XML, since XML requires a great deal of extra padding around the data to create the file format. CSV only needs to be separated by a fixed separator, but it also requires us to format the data according to the file format. Since we have a specific format for each the event types that occurs, and certain differences based on the different fields that comes up for the event.

### XML-formatted output

Figure 20 describes how much of the processor spent being idle during the test. The results are conclusive and depicts that we still had plenty to go on when it came to available resources. This means we could have run this method with many additional users before causing any severe performance issues on the server.

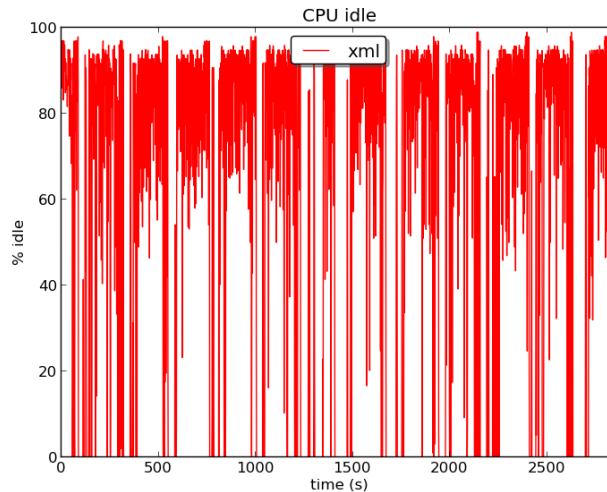


Figure 20: Percentage of time spent idle when using XML

The code for implementing the XML method in Syslog-NG is listed in code listing D.9. This code appends for each entry received from a user, a XML-formatted section to the XML-file for the user. This XML output method had to take into account any values that might be missing from the field and parse it according to our fixed file format before storing into the folder `"/var/log/belt/[user id]"`.

### Unindexed database

Figure 21 describes how much of the processor spent being idle during the test. The results are not conclusive, though it is more idle than the indexed database. We would for certain be able to add more users, but it would be much smaller increase than with the CSV, RAW and XML formats. The figure depicts a medium percentage of time spent idle which indicates that we had approximately 40% to go on before we reached this methods limit. Even though there's, 40% left would we not be able to add too many additional users before it reached its limit.

The unindexed database is created using the MYISAM engine without declaring any relations or primary keys in the table or constrictions. We have created the database using the a version of the SQL-script in listing D.5 of appendix D. The unindexed SQL script differs slightly from our indexed SQL script, 6.3.6. The difference is that the unindexed script does not declare any indexes at the bottom of the script. These last lines of the script declares indexes that has to be created for each row in the table. See listing D.5, in appendix D. The SQL script generates the database shown in figure 22.

We created a SQL procedure common for both the unindexed and indexed databases, see listing D.4 in appendix D. The procedure takes a fixed amount of parameters which is written by Syslog-NG to a FIFO file. Then a bash script, see listing D.11, iterates over



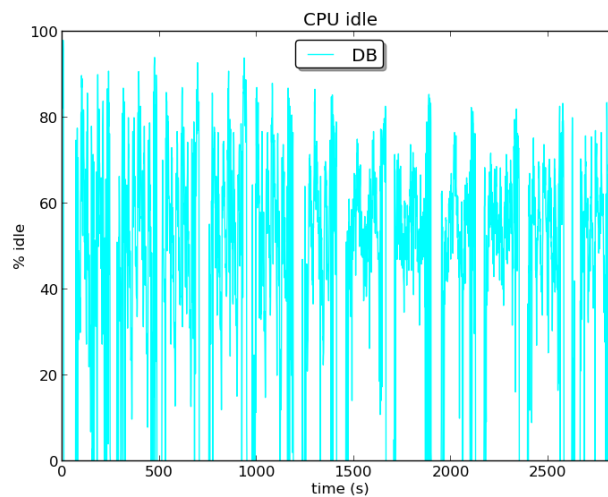


Figure 21: Percentage of time spent idle when using database

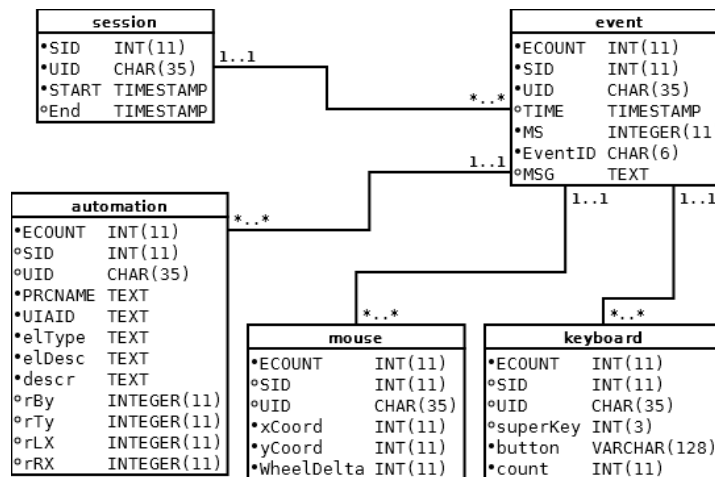


Figure 22: ER-model of our database system

the file and for each line it reads it sends the line to the MySQL server. The MySQL server then runs the procedure and inserts it into the corresponding fields in the tables.

### Indexed database

The indexed database, see listing D.5 in appendix D, is exactly like the the unindexed database with the exception of declaring certain indexes at the end of the script. The reason we chose the indexes we did, was because these fields were the ones we were going to use later on when analyzing our database storage. These fields would help us to correlate the captured data as well as perform quicker searches. We created an index on all tables on the event counter(ECOUNT), the session counter(SID) and the user id(UID). With these three indexes throughout the database would we be able to correlate all tables to identify each row within the table. We also created an index on the timestamp since this would allow for us to search quicker based on time signatures. For the keyboard table we created an index on the button value so we could easier search for the value

based on a buttons value. For the "automation" table we created an index on the process name which made us able to more quickly correlate events within an application. We also created some additional indexes both for us to use them to correlate events, and because we wanted to to have a worst case scenario which required some additional indexes.

Figure 23 describes how much of the processor spent being idle during the test. The results are conclusive and depicts that we reached well above the user limit for our server. The low percentage of time spent being idle clearly indicates this.

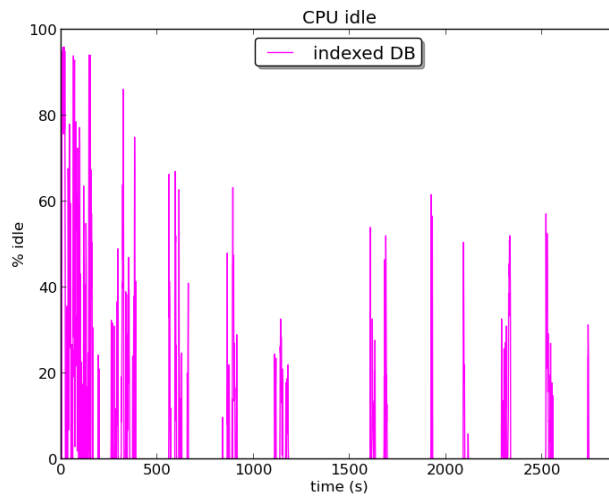


Figure 23: Percentage of time spent idle when using indexed database

## 7 Privacy

In this chapter we discuss the different privacy challenges that we have discovered during the development of BeLT. We also discuss the probable privacy concerns that may occur during a future implementation of BeLT in a research setting, as well as an application in a real world implementation.

During our development we discovered a number of privacy concerns which we implemented mitigating actions towards. Our main concerns have been to keep the users anonymity and to ensure confidentiality and integrity of captured data.

After the algorithm for authenticating users is finished BeLT is planned to run locally on computers and continuously authenticate users. In this scenario BeLT will never send the logged data to a server for storage, but rather locally work together with another application (built for authentication).

### 7.1 Anonymity of the user

When we are capturing data we are also capturing information that can identify the person behind the data. To anonymize the real identity of the person behind the data BeLT generates and gives each user a random, unique ID. This ID is generated in two different ways – the primary method is to generate the ID by selecting a 128 bit unique identifier from the OS which is unique for windows OS-es along with the users username unique to the local computer. This combination ensures that it is a unique value. This concatenated value is then hashed into a 128 bit value, using the MD5 hashing algorithm. This is again encoded as base64<sup>1</sup>.

When doing this it is possible that there could occur a hash name collision. The problems related to this occur when we store information on the server. Identical IDs will cause information from multiple users to be stored as a single user, append data to an already existing file or corrupt the existing file. All cases will render the research data useless for analysis.

To find the likelihood of a collision, we should not calculate based on the number of possibilities alone. We will get a more realistic likelihood of a crash, if we use the birthday attack. Here we use an approximation, since calculating the exact value, is very time consuming and not necessary for us. All calculations have been approximated with the following function:  $1 - e^{-n^2/(2m)}$  where  $n$  is the number of hashes and  $m$  is the number of possibilities ( $2^{128}$ ) Our calculations show that we need to generate about  $2^{64}$  hashes on average before we get a collision. With thousand, or even millions of users, the chances of a collision are infinitesimal<sup>2</sup>.

After the ID is generated it is stored in the users profile to keep it persistent. The user profile file in BeLT is stored in the AppData-folder, all users on Windows has their own AppData-folder and makes it possible for BeLT to distinguish between different users on one computer.

---

<sup>1</sup>Slightly modified so it's safe as a filename

<sup>2</sup>Using Python with 53-bits precision

If our primary method fails (we don't get access to the system ID), we generate an ID by grabbing the current username and append a randomly generated 15 character string to it. This string is then hashed into a 128-bit value, which is encoded to base64 and stored in the current users profile. In this case also, there is a very low probability for ID collisions.

## 7.2 Confidentiality

Since we are capturing and transmitting sensitive and personal information across publicly open networks we had to ensure the confidentiality of the data, as required by the Norwegian law on privacy[57]. To ensure the security and confidentiality of the data when transmitting it to the server we've encrypted the communication between the server and the client.

The system uses the TLS protocol to encrypt the communication lines between the server and the client. The encryption is based on a TERENA certified certificate generated from a 2048b long key using OpenSSL. By using encrypted communication we ensure that an adversary will not be able to read the data during transmission as part of an eavesdropping attack, unless he has the certificate used to decipher the communication.

Another way we keep the users privacy, is that we give the user a choice to store the captured data locally before sending it. The user has the option of manually excluding timeframes before sending the locally stored file. Even though this is a tedious way of filtering out unwanted information, it shows the potential of how we can filter out data.

In future development it should be implemented additional methods for removing data based on what application that's been used. I.e a user should be able to specify whether they want to remove the data gathered while writing an e-mail or a browsing the web. Another method should be to find and remove usernames, passwords, account numbers and similar sensitive information. I.e one could interface BeLT with password managers in web browsers and implement the ability of storing unwanted text strings that should be removed from the capture before transmitting it to the server.

Since we are capturing everything the user is typing, we wanted to minimize the problem for the user to capture very personal information like passwords. We are mitigating this by continuously checking whether the user is currently typing in a password field. When this occurs we disable the logging of keypresses and notify the user by changing the icon in the system tray. Because of this we help the users by automatically removing their passwords and therefore securing what is captured from being very easily abused by a third party. Even though we remove passwords as best as possible we are not able to remove other personal information like usernames and account numbers automatically. Because of this we have implemented the possibility of manually pausing and resuming BeLT upon request by the user and thereby stopping the logger from capturing data.

## 7.3 User awareness

Before capturing the data, we have to make sure that the user is aware of and accepts the purpose of and functionality that BeLT provides. To do this the user is informed of BeLT's functionality and purpose through the EULA(End User Licence Agreement) we have implemented an EULA in the MSI installer. The user is then forced to accept our EULA before they can install BeLT.

The EULA informs the user of what information we are capturing and transmitting,

how it is stored/transmitted, information about BeLTs functionalities and disclaimers to avoid some legal and non-legal issues. I.e we have a disclaimer stating that there is no warranty on the application and that the user is responsible for any use of the application.

In order for a EULA to be as helpful as possible it is important it is easy to read, easy to understand, but most of all is provided in a format that can be easily accessed and managed by the user. The EULA should contain a disclaimer and a copy of the programs licence agreement if its been developed under a specific licence. Any restrictions on the application must be clearly specified within the disclaimer or heading.

Beside the EULA the distributor of the application should have an easy to navigate website explaining the application step by step

In order to assist with informing the user of what we are capturing, we have made a GUI for BeLT. The GUI can be accessed from the system tray icon, and it can display the information that BeLT is capturing in real time. By default, it displays nothing and does not show up, because we've made BeLT to run unobtrusively on the computer. The curious user however, might want to take a look and see what information we are logging, because of this we have made possible to view the data from BeLT.

In addition to the EULA and the GUI we have also created a user manual which explains BeLT and how to use it, appendix B. This guide explains BeLTs interfaces, indicators and settings. It also provides the user with a list of known problems that can occur along with what the reason for the error is and how to mitigate it.

So in summary we have implemented several features to raise user awareness about BeLT and how it functions. In the future it would be a good idea to create a website for BeLT which is the common information channel for BeLT. Here one could explain BeLTs functionalities, purpose and use better and in more detail than in the EULA. This website would also serve as a single point for the public to get information from as well as contain a system for error reporting.

## 7.4 Abuse by authorized personnel

The server needs to be administered by someone, some users will be granted legitimate access on the server with access rights to various degrees. We recommend that abuse by these users should to be mitigated with both policy and technical solutions.

A policy should be in place to state what is considered legitimate use – who has access, their level of access, how the data is processed, how it should be transmitted and projected. The policy also has to state the consequences for when these rules are disobeyed. As part of handling a violation of the policy there should be a contingency plan on how to handle at least the following cases – illegal distribution of data, illegal interaction with data, sabotage of server and manipulation of stored data. The policy should also describe when manipulation of the stored data is allowed, how it should be done and whom is allowed to do it.

For the technical solutions we recommend that there is implemented appropriate access control list(ACL) on all resources and a system event logging service on the server. As a general rule, users should be granted the least amount of access necessary to avoid letting users have access to sensitive resources. By implementing a logging system and setting the correct access rights, access violations will be easily detectable and possible to discover since the logging system would store the violations.

The privacy law (Personopplysningsloven[8]) §13 states that one has to provide ade-

quate integrity of the stored information. To validate the integrity of the stored files, we recommend that one calculates the hash of a finished session and store it securely. This way one can later on validate the content of the file by comparing the to hashes and see whether they're equal. If they are equal, we now that it has not changed, unless someone has successfully replaced the file with something that creates the same hash value.

In addition server updates should be tested on a separate system before being implemented on the active system. This will minimize the amount of issues caused when a new update, feature or configuration is to be implemented.

Since NISlab will use the data stored on the server to develop the authentication algorithm, there has to be policies in place that describes how the administrator can handle the data and what data should be transmitted as output. We recommend that any output from the analysis software shouldn't contain any actual data, but instead only displays the result of the analysis. The development of the algorithm should also be done on the server to avoid moving data from the server to the developers computer.

The most effective mitigative actions is to ensure adequate access control throughout the system and its files, along with a logging system. This is because adequate access control ensures that no user has access to privileged information. The logging system will then work as a monitoring service that detects access violations and can show whom is violating their access rights. Which will back-up the legality of applying consequences for policy violations.

## 7.5 Abuse by un-authorized personnel

In general there are many attack vectors to a system. Our technical countermeasures to reduce these vectors are – TLS encryption, code signing and physical security of our server.

The encryption makes it impossible for an adversary to perform a man in the middle attack(MITM) to gain access to the server or read the data during transmission. It might be possible to take leverage software bugs to gain elevated privileges on the client, thus gaining the data before it is being encrypted. On the other hand the server might be compromised due to software bugs which could grant an attacker access to the server and its data. Since this is more than a likely scenario it is important to implement good routines for keeping the server up to date with security patches.

The server itself can pose a risk if it is not properly secured against un-authorized access, in the physical sense. We recommend the server to stay in a physical safe and secure location.

Another attack vector is impersonation. An attacker may create a similar software to impersonate BeLT, but instead transmit the data directly to the attacker. This is however, not a likely scenario. Even if our adversary has the resources to perform this attack, it is still to some degree mitigated. It is mitigated because we are signing our code by using certificates certified and trusted by TERENA[40]. This way our installer is digitally signed and whenever the UAC dialog appear it states the publisher of the application, "Høgskolen I Gjøvik". This creates trust between the user and the application that it is in fact created by the correct persons and trusted by larger corporation. The client also validates the server with a certificate chain, distributed with BeLT.

Since this system is a potential target it is important that it is implemented technical solutions to secure the virtual perimeter around the server. This should incorporate a

secure zone behind a firewall that uses ACLs to control access to the server behind the firewall. These ACEs should describe both the host and what protocol it is allowing. The firewall should incorporate a whitelisting that denies all traffic beside specified entities. These entities should be a range of fixed IP-addresses used by the system administrators for SSH access to the server, and all users connecting to the port used for receiving data to the Syslog-NG service.

## 7.6 Transparency of logged data

When capturing and storing personal information the user may want to see the data that is captured. We recommend that the user should be completely anonymous, even to its own data – by doing this we make sure that none of the users can try to acquire other users data. This might give the user wrong impression because we do not want any secrecy in what we are doing – rather we want to do everything possible to keep the users information safe.

We have implemented two things to ensure a transparency of data.

1. A representation of the raw data within the application to see what is captured.
2. Option to locally store and review BeLT sessions before sending them to the server.

The purpose of this is to establish trust between the users, owners and administrators of the system where the data is stored – keeping the users more satisfied.

To help the user avoid sending unwanted data we have added the option of storing data locally in a folder on the computer before it's sent. By doing this the user can view the locally stored sessions with timestamps and see what information that is stored – and avoid sending the file before excluding sensitive data from the file.

BeLT should in future versions also have the ability to filter out any event generated as part of an application – i.e Microsoft Office, Outlook and alike. By doing this, BeLT could possibly remove all events that corresponds to a personal mail or online banking session. We believe it should also be possible to remove passwords, bank account numbers and other information based on filtering rules. If for example the users could add their username, password and bank account number to a rule set in the application and later on before sending data it would remove those events from the log. This would help the user in keeping the trust that their passwords and username is not being compromised.

## 7.7 Storage of data

When storing personal information the privacy law (Personopplysningsloven) [8] has three paragraphs covering requirements to the storage and safekeeping of the data. §13 states that the implemented security countermeasures must be documented, and that there must be a satisfactory level of information security in place. These countermeasures must ensure that confidentiality, integrity and availability is upheld when processing, collecting and storing personal information. This involves avoiding to corrupt any data during storage, processing or transmission. It also involves securing the server against both authorized and unauthorized personnel, see section 7.4 and section 7.5 for our proposed methods of securing the server against these type of attacks.

All the data should have a fixed lifespan. Since the data is going to be used in a research project, it is hard for us to determine, how long the data should be stored. But it should be set before the project is started and the user should be informed about this.





## 8 Conclusion

### 8.1 Achievements

One of the main objectives we didn't have a solution for, was how to retrieve information about how the user interacts with software. Previous work provided little information for how to do this and we found no generic method to collect this type of information. We investigated possible solutions and came up with User Interface Automation, which turned out to be a good solution.

Another challenge for us was to develop a large application that is going to be distributed to multiple users, and be further developed after we are finished. This is not something that any of the group members had done before and we needed to take a more professional approach to system development than we had taken before. We did this by working in fixed increments with regular meetings with the employer, using bug tracker to keep a record of progress and bugs. We also set up a complete development environment in which we could work, and cooperate efficiently.

One unexpected challenge that occurred during development was the transmission between the client and the server. We had to find a secure way to transmit all the logs to the server. First we identified Syslog-NG as a potential program we could use on the server. Syslog-NG relies on the Syslog protocol, which satisfied all our requirements for transport. On the client side we didn't find any reliable implementation that we could use, therefore we had to program the transmission component ourselves, following RFC5424.

The final application satisfies the requirements quite accurately. Almost all of the decisions we have made along the way has provided good results. From a functional standpoint the application gathers and stores the appropriate information. From the user perspective, the application is unobtrusive and everything can be done without the users interaction.

### 8.2 Requirement specification and results

Requirement specification and our results	
Requirement	Status
Capture when keys are pressed and released	Completed ✓
Capture mouse interaction	Completed ✓

*Continues on the next page*

Requirement	Status
Distinguish different HW components used (external mouse, touchpad etc.)	The method we first used to detect different hardware worked in some scenarios, but in other scenarios it provided completely wrong results in other again it provided no results. It seems like all mouse hardware is grouped in as one logical mouse on some systems, which means that we probably need to create a device driver to accurately gather this type of information. <span style="float: right;">✗</span>
Capture peripheral equipment and their status	Completed. <span style="float: right;">✓</span>
Capture how the user interacts with software	What information is captured and the type of data we store has been developed throughout the project, until the amount of information was detailed enough. We also found a generic way to gather information, so that includes any applications that was set as a minimum. <span style="float: right;">✓</span>
Correlate events	We have developed some simple rules to decide if two events are connected or not. See section 5.4.2 for a discussion of how we do that and possible wrong correlations. <span style="float: right;">✓</span>
Timestamp with millisecond accuracy	Timestamps with millisecond accuracy $\pm 16$ ms. After a talk with the employer, this was satisfactory, see separate section about time granularity 6.1.3. <span style="float: right;">✗</span>
Ability to detect lost packets in retrospect	Each event has a counter and each session has a start and stop message. If we lose packets in the middle or at the start, we know how many packets has been lost, if we lose packets at the end, we only know that packets has been lost. <span style="float: right;">✓</span>
Secure transmission of data	All data is transmitted over SSL/TLS, which provides both security and integrity. <span style="float: right;">✓</span>
The application should at minimum run at Windows 7 and later	Because of libraries used, the user need at least Windows XP SP 3, which corresponds with the requirement. <span style="float: right;">✓</span>
Application signed by certificate authority	The application is now digitally signed with a certificate, provided by HIG, and the process for doing this has been detailed, so the employer can sign the code with their own certificates. <span style="float: right;">✓</span>
Implement mitigative measures to hinder a third party to pose as another user	All the IDs are generated randomly and hashed with MD5. Each ID is considered a secret and is only sent encrypted over the network. <span style="float: right;">✓</span>

*Continues on the next page*

Requirement	Status
The program should run unobtrusively	By default, the program starts and stops everything when the user power on and power off the computer. The user has the option to turn this off. The program resides in the system tray with minimal obstruction to the user, while still visible. ✓
GUI design	There was no hard requirement for how the GUI should look. After discussions and programming, we came to a solution that was simple and provided the basic functionality and hides all advanced features. For a discussion of the design and how it progressed, see chapter 4.4. It does contains all the elements that were given as requirements. ✓
The loss of logged events can't be greater than 1%	We have not tested this actively, but have never experienced any loss of events, so we are quite sure we are within this boundary. ✓
Estimate total amount of possible users	We have given a moderate estimate based on our limited test. Since we didn't saw any problems with a couple hundred users, these estimates are higher than what we expect to observe in the beginning. ✓
Anonymous, unique and persistent identifier	Each user gets an ID based on computer properties and the username. This ID will be the same for that specific user on that specific computer, but will be different for another user on the same computer. ✓

Table 15: Which requirements we have fully completed and which we have not fully completed, for some reason. ✓ means completed, and ✗ means not completed.

Table 15 shows a summary of which requirements we have fully completed and which requirements has not been solved for some reason.

The functionality of the application follows closely with the requirement specification. This also follows from the fact that the requirement specification was not static throughout the project. Several parts where undefined in the beginning and had to be defined as we progressed.

The main part of the application is what we store about each user session. There was no right or wrong way to do this. We had to find a method that we thought would work and decide with the employer. We found a method that collects information about any program that the user interacts with, after a discussion with the employer, we agreed upon this method. Collection of how the user interacts with input devices has been pretty static since the beginning, but has progressed some under meetings with the employer.

The format for the data, had to be in a CSV file format. Exactly how this format should be written changed over time, but we all agreed on a final format, which is what is used in the final application.

## Tasks not in requirement specification

During development we also discovered several features that was not part of the requirement specification, but would be useful.

CSV files are easy to read and analyse when you know what the data is like, but for future work, we came to the conclusion that XML would be a more complete format for this type of data. We implemented the possibility for both CSV and XML in the final application.

Storing this type of information in separate files throughout the system can get complicated fast. When you have large amount of information, you also need a fast way to retrieve certain properties. Databases are excellent for this purpose, therefore we also implemented a method for inserting all data into a database. Since this couldn't be done directly from Syslog-NG (see section 6.3), we created a cron-job which could run every night (when very few people are using the system).

One of the problems we saw with this type of application, is privacy, and what users would think about this type of information gathering. We are open and honest with everything we collect, but some users may still have a problem. Therefore we also implemented a method to store the files locally first, before sending it to the server. The user also has the options of filtering out certain time frames. The method we used here is quite limited and provides very little functionality, but it serves as a prototype for what is possible, in section 8.3 we discuss more ideas about this.

Automatic update is a central part of almost any modern application, we have implemented this functionality, but since the application will not be in active development after we are finished with the project, it has been turned off.

To test the client-server communication, we used a temporary server. We where supposed to set up and configure the development server before the report was handed in, but the server was not ready, as of May 15.

## 8.3 Future Development

This application serves as a prototype to see if it can be used in this type of biometrics. It was not supposed to be finished now and we hope it will be developed more in the future. During the development process this has always been considered, so we have tried to document all the steps we have taken and all the decision we have taken, from the server, to distribution to the client application.

Since this can be a self-contained program in the future, designed for authentication, we have also implemented patching functionality.

A device driver can gather more specific information in a couple of cases; detect different hardware used is one of the problems we think can be solved with a device driver. More accurate timestamps might be possible with a driver, but you would probably come into the same problems as we did when analyzed timestamp accuracy, see 6.1.3.

The information we now gather is based on what we think will be useful. It is not unlikely that this will change in the future when developing the algorithm for authenticating users. We only gather a small subset of all the available information.

Ont thing that should be done in the future is; a more rigorous test to get the best performance out of the server and to get a better understanding of how many users the final server will be able to handle.

We have implemented a function for sending locally stored file to the server, the user

can then filter out certain time frames. It might be difficult for the user to know what the filter out when you only have time frames to work with, so a way to visualize the data and filter out everything the user did in a given window would be much more useful.

Another way to filter out data, in a way that more people are willing to participate in the experiment is to let the user filter out given words or application beforehand.

AppMonitor [5] reduced privacy concerns by not storing key events that could be used to reveal what text they typed, which is another group of events that could be excluded.

Another part that might make it more fun for the user is to store statistics for much longer time. We can for example store how far the mouse has travelled or how many words have been written while the program has been running.

## 8.4 Alternatives

We made several decisions regarding how the application should work, the most substantial one was probably the decision to use User Interface Automation (UIA) to gather information about how the user interacts with software. Another option here was to use Microsoft Active Accessibility (MSAA), which was the predecessor to UIA. It doesn't look like that would have changed much, but we did have to use MSAA for some type of events. We do think that UIA was the better option, both for compatibility in the future and maintainability as UIA is easier to navigate.

When choosing a protocol for transmission over the Internet, we had to keep a couple of things in mind, it had to be secure and fast. One thing we considered early was that we might be able to use an already existing product on the server, that would both make our job easier and make our application more flexible. We came to the conclusion that Syslog-NG would satisfy all the necessary requirements. After this was chosen we had to use the Syslog protocol for transmission which satisfied our requirements to security. By doing this the application on the server can easily be substituted to another application that support the Syslog protocol. This makes our application more universal. The main disadvantage with using Syslog as the protocol is that we need to send each event separately, regardless of size, this causes more overhead.

We quickly chosed to use OpenSSL as our library for TLS communication, this was because we wanted to avoid compatibility issues with the Linux server we had. This decision has not caused any problems, the only downside we can see, is that the libraries has to be distributed by us. This is not desirable, if OpenSSL releases an update, our application should also be redistributed with the new libraries. The only reason to do this, is to fix vulnerabilities, since our application only talks to servers controlled by the development team, it is not a big problem.

When creating the GUI we had to decide which library to use, several are possible, our final choice was MFC, which we chosed without much discussion. The GUI is a small part of the application and this application will never be cross-platform because of all the other libraries used in the application. Therefore we found that MFC will provide a decent solution to our problem.

## 8.5 Evaluation of group work

The group has worked well together. All of us has worked on projects together before, so there where no surprises there. We mostly agreed on decisions, if we didn't agree it was easily solvable.

We had two meetings a week, where we discussed what we had done, any problems and what each of us was going to do in the next couple of days. The length of these meetings varied greatly on how much we had to discuss.

### **8.5.1 Work process**

We decided to work in increments of two weeks before we had a product that could be delivered for testing. This gave us some leeway where we could test out different things without having to worry about the next delivery. For us this process worked very well, especially since there was a lot that was unclear when we started the project and the requirement specification was further developed along the way.

## Bibliography

- [1] Ernesto Arroyo, Ted Selker, and Willy Wiy. Usability tool for analysis of web designs using mouse tracks. *CHI EA '06 CHI '06 Extended Abstracts on Human Factors in Computing Systems*, pages 484–489, 2006. <http://dl.acm.org/citation.cfm?id=1125557>.
- [2] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. *WWW '06 Proceedings of the 15th international conference on World Wide Web*, pages 203–212, 2006. <http://dl.acm.org/citation.cfm?id=1135811>.
- [3] Florian Mueller and Andrea Lockerd. Cheese: tracking mouse movement activity on websites, a tool for user modeling. *CHI EA '01 CHI '01 Extended Abstracts on Human Factors in Computing Systems*, pages 279–280, 2001. <http://dl.acm.org/citation.cfm?id=634233>.
- [4] Urmila Kukreja, William E. Stevenson, and Frank Ritter. RUI: Recording user input from interfaces under windows and Mac OS X. *Behavior Research Methods*, 38:656–659, November 2006. <http://link.springer.com/article/10.3758%2FBF03193898>.
- [5] Jason Alexander and Andy Cockburn. AppMonitor: A tool for recording user actions in unmodified windows applications. *Behavior Research Methods*, 40:413–421, Mai 2008. <http://link.springer.com/article/10.3758%2FBRM.40.2.413>.
- [6] A. Garg, S.Vidyaraman, S. Upadhyaya, and K. Kwiat. USim: A User Behavior Simulation Framework for Training and Testing IDSes in GUI Based Systems. *Simulation Symposium*, 39, 2006.
- [7] Hugo Gamboa and Vasco Ferreira. Widam - web interaction display and monitoring. In *5th International Conference on Enterprise Information Systems, ICEIS'2003*, pages 21–27. INSTICC Press, 2003.
- [8] Lovdata. Personopplysningsloven. <http://lovdata.no/all/h1-20000414-031.html>. Accessed: 18.03.2013.
- [9] The Norwegian Data Protection Authority. <http://datatilsynet.no/English/>. Accessed: 15.04.2013.
- [10] Microsoft. UI Automation (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ee684009>. Accessed: 09.01.2013.
- [11] Microsoft. Microsoft Active Accessibility and UI Automation Compared (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/dd561918>. Accessed: 09.01.2013.

- [12] Microsoft. UI Automation Clients Overview (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ff625909>. Accessed: 09.01.2013.
- [13] Microsoft. Subscribing to UI Automation Events (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ee671220>. Accessed: 09.01.2013.
- [14] Microsoft. UI Automation Events Overview. <http://msdn.microsoft.com/en-us/library/windows/desktop/ee671221%28v=vs.85%29.aspx>. Accessed: 11.04.2013.
- [15] Microsoft. Hooks (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ms632589>. Accessed: later.
- [16] Microsoft. KeyboardProc callback function. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644984\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644984(v=vs.85).aspx). Accessed: 13.04.2013.
- [17] Microsoft. LowLevelKeyboardProc callback function. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644985\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644985(v=vs.85).aspx). Accessed: 13.04.2013.
- [18] Microsoft. LowLevelMouseProc callback function. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644986\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644986(v=vs.85).aspx). Accessed: 13.04.2013.
- [19] Microsoft. MouseProc callback function. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644988\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644988(v=vs.85).aspx). Accessed: 13.04.2013.
- [20] Microsoft. Certification requirements for windows 8 desktop apps. <http://msdn.microsoft.com/en-us/library/windows/desktop/hh749939.aspx>. Accessed: 07.03.2013.
- [21] Microsoft. Windows software development kit (sdk) for windows 8. <http://msdn.microsoft.com/en-us/windows/desktop/hh852363.aspx>. Accessed: 08.03.2013.
- [22] Microsoft. Testing your app with the windows app certification kit. <http://msdn.microsoft.com/en-us/library/windows/apps/hh694081.aspx>. Accessed: 07.03.2013.
- [23] Erik Fløisbonn. Integrating conduit with windows installer. <http://hdl.handle.net/10852/10089>, 2009.
- [24] Microsoft. Windows installer package. <http://technet.microsoft.com/en-us/library/cc978328.aspx>. Accessed: 13.04.2013.
- [25] Microsoft. Windows installer. [http://msdn.microsoft.com/en-us/library/cc185688\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/cc185688(v=vs.85).aspx). Accessed: 13.04.2013.
- [26] Microsoft. Orca.exe. [http://msdn.microsoft.com/en-us/library/windows/desktop/aa370557\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa370557(v=vs.85).aspx). Accessed: 13.04.2013.



- 
- [27] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with coding theory*, chapter 9, pages 249–250. Pearson Prentice Hall. ISBN: 0-13-186239-1.
- [28] R. Gerhards. The Syslog Protocol. <http://tools.ietf.org/html/rfc5424>. Accessed: 04.03.2013.
- [29] C. Lonvick. The BSD syslog Protocol. <http://www.ietf.org/rfc/rfc3164.txt>. Accessed: 25.04.2013.
- [30] Eric Fitzgerald, Anton Chuvakin, Bill Heinbockel, Dominique Karg, and Rafael Marty. *COMMON EVENT EXPRESSION (CEE) OVERVIEW*. MITRE, 1.0 edition. [http://cee.mitre.org/docs/Common\\_Event\\_Expression\\_Overview.pdf](http://cee.mitre.org/docs/Common_Event_Expression_Overview.pdf) Accessed: 25.04.2013.
- [31] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. <http://tools.ietf.org/html/rfc4346>. Accessed: 15.04.2013.
- [32] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. <http://tools.ietf.org/html/rfc5246>. Accessed: 04.03.2013.
- [33] Lisa Feigenbaum. Documenting your code with xml comments. <http://msdn.microsoft.com/en-us/magazine/dd722812.aspx>. Accessed: 29.04.2013.
- [34] Dimitri van Heesch. Doxygen: Documentation generator. <http://www.stack.nl/~dimitri/doxygen/>. Accessed: 23.03.2013.
- [35] Microsoft. Caching UI Automation Properties and Control Patterns (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ee684019>. Accessed: 09.01.2013.
- [36] Microsoft. Understanding Threading Issues (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ee671692>. Accessed: 09.01.2013.
- [37] Microsoft. Using UI Automation for Automated Testing (Windows). <http://msdn.microsoft.com/en-us/library/windows/desktop/ee684083>. Accessed: 09.01.2013.
- [38] Microsoft. Automation Element Property Identifiers. [http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017\(v=vs.85\).aspx#uia\\_ispasswordpropertyid](http://msdn.microsoft.com/en-us/library/windows/desktop/ee684017(v=vs.85).aspx#uia_ispasswordpropertyid). Accessed: 14.04.2013.
- [39] Jordan Russel. Innosetup. <http://www.jrsoftware.org/isinfo.php>. Accessed: 08.03.2013.
- [40] Terena. <http://www.terena.org>. Accessed: 15.04.2013.
- [41] BalaBit IT security. Syslog-NG OSE. <http://www.balabit.com/network-security/syslog-ng/opensource-logging-system/ov>. Accessed: 04.03.2013.
- [42] OpenSSL. OpenSSL. <http://www.openssl.org/>. Accessed: 04.03.2013.
- [43] OpenSSL. OpenSSL License. <http://www.openssl.org/source/license.html>. Accessed: 04.03.2013.

- [44] James Kovacs. Psake: build automation tool. <https://github.com/JamesKovacs/psake>. Accessed: 24.03.2013.
- [45] Microsoft. Overview of rule sets used by visual studio. <http://msdn.microsoft.com/en-us/library/dd264925.aspx>. Accessed: 30.04.2013.
- [46] Microsoft. Mixed recommended rules rule set. <http://msdn.microsoft.com/en-us/library/hh748337.aspx>. Accessed: 30.04.2013.
- [47] Microsoft. All rules rule set. <http://msdn.microsoft.com/en-us/library/dd264971.aspx>. Accessed: 30.04.2013.
- [48] Mozilla Foundation. Bugzilla. <http://www.bugzilla.org>. Accessed: 04.03.2013.
- [49] Microsoft. KBDLLHOOKSTRUCT structure. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644967\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644967(v=vs.85).aspx). Accessed: 10.05.2013.
- [50] Microsoft. Windows Time. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms725496\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms725496(v=vs.85).aspx). Accessed: 13.04.2013.
- [51] Microsoft. QueryPerformanceCounter function. <http://msdn.microsoft.com/en-us/library/windows/desktop/ms644904>. Accessed: 31.03.2013.
- [52] Microsoft. QueryPerformanceFrequency function. <http://msdn.microsoft.com/en-us/library/windows/desktop/ms644905>. Accessed: 31.03.2013.
- [53] Microsoft. timeBeginPeriod function. [http://msdn.microsoft.com/en-us/library/windows/desktop/dd757624\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd757624(v=vs.85).aspx). Accessed: 13.05.2013.
- [54] Microsoft. About Messages and Message Queues. <http://msdn.microsoft.com/en-us/library/windows/desktop/ms644927>. Accessed: 31.03.2013.
- [55] Zoltán Pallagi. syslog-ng performance tuning. <http://pzolee.blogspot.com/2011/02/syslog-ng-performance-tuning/>. Accessed: 31.03.2013.
- [56] Sebastien Godard. System Activity Report(SAR). [http://sebastien.godard.pagesperso-orange.fr/man\\_sar.html](http://sebastien.godard.pagesperso-orange.fr/man_sar.html). Accessed: 15.04.2013.
- [57] Lovdata. Personopplysningsloven §13. <http://lovdata.no/all/h1-20000414-031.html#13>. Accessed: 15.04.2013.

# BELT: Behavior Logging Tool System manual

**Robin Stenvi** robin.stenvi@hig.no  
**Magnus Øverbø** magnus.overbo@hig.no  
**Lasse Tjensvold Johansen** lasse.johansen@hig.no



(Behaviour Logging Tool)



(Norwegian Information Security laboratory)

May 15, 2013

---

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Script overview . . . . .	3
1.2	Tools overview . . . . .	4
<b>2</b>	<b>Application</b>	<b>6</b>
2.1	Technology overview . . . . .	6
2.2	Graphical User Interface . . . . .	8
2.3	Message map . . . . .	9
<b>3</b>	<b>Server</b>	<b>10</b>
3.1	Server overview . . . . .	10
3.2	Syslog-NG . . . . .	10
3.3	Syslog protocol . . . . .	19
3.4	Database . . . . .	20
3.5	Export XML to XML . . . . .	33
3.6	Export database to CSV . . . . .	33
3.7	Export CSV to CSV . . . . .	39
3.8	Bugzilla . . . . .	41
3.9	Import and export of Bugzilla databases . . . . .	43
<b>4</b>	<b>Software Distribution</b>	<b>45</b>
4.1	Deploy.ps1 - main script . . . . .	45
4.2	Windows Installer XML toolset(WiX) . . . . .	45
4.3	Codesigning . . . . .	47
4.4	WiX build-script . . . . .	48
4.5	WiX configuration file . . . . .	50
4.6	WiX patch configuration file . . . . .	53

# 1 Introduction

## 1.1 Script overview

`/deploy.ps1` Is the main script for building binaries and new software releases. We have scripted `deploy.ps1` so that it will check if you have all dependencies installed before it begins to execute its actions.

It depends on: psake, signtool, openssl, subversion and doxygen

Listing 1: Parameter overview

```

1 #####
2 # Parameter overview
3 #####
4 # -b build type "patch" or empty
5 # -l Specifies whether to deploy to upload documentation and
6 # source code to repo. 0 deploy to repo and 1 to avoid.
7 # -p platform specification "x64" or "x86"
8 # -n The new version number"
9 # -o The old version number (if you make a patch) [x.x.x]
10 # -x The password used for code signing
11 # -s to only compile the executable and generate installer
12 #####

```

Listing 2: Example of all parameters in use

```

1 deploy.ps1 -n x.x.x -x password [-b patch -o y.y.y] [-p x86/
  x64] [-s true/1] [-l upload/0]

```

Listing 3: Example of building an upgrade

```

1 deploy.ps1 -n 1.0.0 -x password

```

Listing 4: Example of building a patch

```

1 deploy.ps1 -n 2.0.0 -x password -b patch -o 1.0.0

```

`/Code/belt_main/buildx64.cmd` This commandlet provides a command using the psake commandlet to build BeLT using the `build.ps1` PowerShell script. This commandlet builds the x64 version of BeLT, and doesn't take any parameters.

`/Code/belt_main/buildx86.cmd` This commandlet provides a command using the psake commandlet to build BeLT using the `build.ps1` PowerShell script. This commandlet builds the x86b version of BeLT, and doesn't take any parameters.

`/Code/belt_main/build.ps1` This is the build scripts that compile the source code using MSBuild.exe to generate the binaries.

- /Code/belt\_main/belt\_installer/wix.ps1** This script builds the MSI or MSP packages. This script has the same parameters as deploy.ps1, and it's called by deploy.ps1. Wix.ps1 is described in detail in its own section.
- /Additional\_Material/dbCronJob/import.py** This script imports data from Syslog-NG XML files into the database and stores it into the correct fields using the MySQL addon for python. In addition to importing data to the database, it can convert the Syslog-NG XML files into UTF-8 encoded XML files with escaped special symbols.
- /Additional\_Material/dbCronJob/export.py** The export.py script grabs the user IDs and session ID of the chosen users and export their data to the predefined CSV format, ready to read by our employer. By providing the "GETALL" parameter the script will automatically retrieve all sessions from all users and export them into the correctly file formatted CSV.
- /Additional\_Material/dbCronJob/DATABASE.sql** This SQL script generates the entire database with fields and indexes on the different tables.
- /Additional\_Material/Code/csv2csv.py** This python script converts Syslog-NG stored CSV files with ISO timestamps to CSV files with correct timestamp formatting.

## 1.2 Tools overview

Overview of tools	
Name	Description
Psake	Tool to define tasks in a PowerShell script and run them from a command shell and from inside Visual Studio
Microsoft Visual Studio 2012	Microsoft's standard IDE for programming applications. Visual Studio Express has limited or no support for MFC, so you need the professional or premium edition.
Doxygen	Software for automatically scanning source code for predefined comments, and generating a documentation in HTML, Latex or RTF format
WiX toolset 3.7	Set of tools for generating MSI installer packages
WinRaR	Tool for archiving the source code
PowerShell	Command line utility in Windows to run our scripting tasks
Latex	Text processing software, that reads latex files and
PDFLatex	Generates a PDF document from a formatted Latex file.
MSBuild.exe	Compiler for Microsoft Windows.
continues on next page..	

<b>Name</b>	<b>Description</b>
SignTool.exe	Application for signing source code and executables
Windows App Cert Kit	Application for testing whether an executable passes the Windows application certification requirements.
Microsoft Project Manager 2013	Microsoft tool to manage our projects development process in the form of a GANTT schema.
System Activity Report(SAR)	Collect statistical information about a hosts system information. Used to test the performance on the server. See separate section in the report.
AccEvent.exe	See what type of events we can get from UI Automation and MS Active Accessibility.
UISpy.exe	See what type of event we can get from UI Automation.
VisualSVN	Integrate Subversion into Visual Studio.
Orca.exe	Investigate how the MSI files are structured.
Apache2	Used to run our webserver services on the server.
MySQL	Used to store our data from the BeLT logger and Bugzilla
MySQLdump	Used to dump all of our data from a database when migrating to a new server
Bugzilla	Bugtracking system which we've been running on our server
Syslog-NG	Server application that can accepts the logs the client generates
python-mysqldb	MySQL database addon for python. Used by the import/export script on the server for database connection

## 2 Application

### 2.1 Technology overview

BeLT relies on 6 technologies:

**General Windows programming in C++** : Here are some of the concepts in the Windows API that should be understood:

**Hooking:** We use hooking to capture key presses and mouse interaction. We don't use a lot of this, so it is relatively simple to learn.

**Messages:** A lot the communication between different parts of the program happens with messages. An understanding of how this works is necessary.

**Callback functions:** We use callback functions extensively throughout the application, a brief understanding of how they work is necessary.

**User Interface Automation:** We use this to gather information about how the user interacts with software. This has a higher learning curve, mostly because of the amount of information.

**Microsoft Active Accessibility:** Some information we are unable to get from the UIA is retrieved here. Used very little in this application and you can probably leave it as it is.

**MFC:** Used to create the GUI, see 2.2. You probably want to look in to this, regardless of what you want to change. It is needed to understand how the application is structured.

**TLS encrypted traffic:** We use OpenSSL to send traffic to the server. This is used on the update mechanism and the logging mechanism, you can probably leave this as is. The client will not create a connection with the server, if the certificates don't match the certificate chain that is supplied.

**Syslog protocol:** We use the Syslog protocol to send data to the server. If you need to change what is logged, you have to understand this protocol. Since we have implemented the protocol ourselves, you need to understand in detail how data is sent.

To get a full grasp of the application, you need to understand all these technologies.

We will also try to go through some likely scenarios of small things that you might want to change, here you don't need the whole picture.

**Server address:** All the server configuration is gathered from the file settings.ini. You can change this without knowing anything about the application, but then the application has to be redistributed. You also need the correct certificate key-chain, which currently is **CA-chain.pem** and



is placed in `%programData%\NISLab\belt`. This file is also distributed with the application. The server address should be based on a domain name, which is static, then you avoid all this trouble. The CA-chain.pem was generated with the following command: `openssl s_client -connect server:port -showcerts` Here you will get a certificate chain, everything between `—BEGIN CERTIFICATE—` and `—END CERTIFICATE—` need to be included, in the order you see it there, currently there are four items in the chain. This should be called on the port for HTTPS. All this can be done with the update mechanism, if that is enabled.

**Default settings:** This is settings for how the application should react on first startup. This is stored in the same configuration file as server settings, so you have to do the same process.

**Change the format of data:** If you only want to change how data is structured, you need to change the source code. To change how data is sent to the server, you need to change the appropriate `getFormat*()` functions in `handleData`. Remember to also change it on the server. If you need to change the format of the local CSV file, you need to change the appropriate `getCsv*()` functions in `handleData`.

**Change what is captured:** Data capture consist of 4 main parts:

**Mouse:** `setLLEvent()` in `Mouse` determines what is gathered. If you need more information that we get from the low level hook, you need to make more significant changes. For a discussion of high and low level hooks, see the report.

**Keyboard:** `setLLEvent()` in `Keylogger` determines what is gathered from our low level hook. High level hook also exist.

**Software:** This part consist of 4 subparts: `registerEventHandler()` and `HandleFocusChangedEvent()` in `focusEventHandler`, `registerEventHandler()` and `HandleAutomationEvent()` in `eventHandler`, `registerEventHandler()` and `HandlePropertyChangedEvent()` in `propertyEventHandler`, `registerwinEvent()` and `WinEventProc()` in `myWinEvent`. Both functions in each class have to be changed. To know what to change, you need an understanding of UI Automation and maybe Microsoft Active Accessibility, if you need to change `myWinEvent`. If you store new information you also need to add a function in `Events` and call it from `fillEventInfo()` in `Events`.

**Hardware:** This part consist of 5 separate subparts:

**Input devices:** The functions `manageRawInput()` and `OnRawInput()` in `Cbelt_mainDlg` has been disabled for now.

**Resource usage:** To change how we monitor resource usage you need to change `monitorHWUsage()` in `Cbelt_mainDlg` and the `HWMonitor` class.

**Device change:** To change how we handle device insertion and removal, you need to change `OnDeviceChange` in `Cbelt_mainDlg`.

**Keyboard:** To change what kind of information we gather about the physical keyboard, you need to change `sendKeyboardInfo()` in `Cbelt_mainDlg`.

**Screen information:** To change how we handle information about the physical screen, you need to look at: `MyInfoEnumProc()`, `checkScreen()`, and `HandleFocusChangedEvent()` in `focusEventHandler.c`.

You also need to follow this throughout the application, change the appropriate structures, the `write*()` functions in `handleData` and so on.

**Implement automatic update:** For this you only need to call `check()` in `checkUpdate`. You should also test how it works and make sure that it is correct. The files on the server have the following format: `updateIP,updatePort,updatePortSSL,thisFile,currentVersion,listOfPatchesFile,LoggingServer,LoggingPort`. IP-addresses can also be domain names. The files are files on the server and should contain an absolute path, saying how to reach it from the outside. The list of patches consist of several lines, where each line says the version you need to download this patch and the link to both x64 and x86 patches, like this: `version,x86PatchFile,x64PatchFile`. You need a full path to the files.

**Test / change mouse compression:** If you run the debug version of the application, it will generate two log files. One contains all the mouse movements we saw, the other contains the events we actually logged. If you plot these into our Python script for painting graphs, you can get a general overview of how the algorithm works. You can also use the other script to test new compression values. Since one of the files contain all the movements, you can also use it to test new compression techniques.

## 2.2 Graphical User Interface

By using the MFC library we have access to a vast amount of elements to implement in our application. Buttons, text fields, progress bars, dialogs, and so on. With Visual Studio one can also easily create the GUI. First by creating a new MFC project, which provides the basis of the dialog interface with a couple of buttons and the menu bar.

The initial GUI layouts of buttons, fields, dialogs, menus and alike are defined in the `".rc"` file which specifies all initial GUI elements. These are then rendered on start and creates the initial GUI layout. Further GUI development can either be done by adding buttons in the source code or by using the visual editor in visual studio to add and edit the elements in the `".rc"` file.

use the `DoDataExchange(CDataExchange* pDX)` function in the class to assign the declared MFC object a unique ID from the resource file. What this does is that it will now link the MFC object and the information in the resource

file together, which enables you to dynamically change the information of the object. This is done by inside the `DoDataExchange(..)` declaring the following:

```
DDX_Control(pDX, id_of_resource_object, MFC_object);
```

### 2.3 Message map

When a "Message" is generated and sent by the system/application the message map declared in the application will receive the message. Then the application will act direct the message according to the declared "message map".

An example of declaring a message map is as follows:

```
BEGIN_MESSAGE_MAP( Cbelt_mainDlg, CDialogEx )
    ON_MESSAGE( MESSAGE_MACRO, afx_msg function )
    ON_BN_CLICKED( BUTTON_ID, afx_msg function )
END_MESSAGE_MAP( )
```

The first line states the local class for the message map, followed by the local classes base class. The second line declares what function to run when receiving a message, with a message macro. The third line is to run the function in parameter two when the button referenced as parameter one is clicked. Finally the last line will declare the end of the message map.

The message map is declared in the main source code, while the message map is declared in the header file, by adding the "DECLARE\_MESSAGE\_MAP" at the end of the header. The functions that's part of the message maps must be declared as with the "*afx\_msg*" prefix because of a remnant from the early stages of the MFC library development.

## 3 Server

### 3.1 Server overview

The server is set up with several services and are running the following.

**Apache:** hosts our bugtracker site and deployment site

**MySQL:** contains the data from our bugtracker and captured data from BeLT

**Syslog-NG:** receives data BeLT and stores it. The version used is OSE 3.3 LTS. If you change how the client send logs, you also need to change the Syslog-NG config file which is located in `/etc/syslog-ng/syslog-ng.conf`.

**System Activity Report(SAR):** collects statistical data about the servers performance under testing (not a continuous service)

**Bugzilla:** is the bugtracking software, it has its own database and runs as an entity under `/var/www/`

### 3.2 Syslog-NG

Syslog-NG is used as the server that accepts the logs from the client. It uses the Syslog protocol and it supports UDP, TCP and TLS. We use TLS for all communication.

To install Syslog-NG at an Ubuntu server, all you need to do is:

```
1 apt-get -install syslog-ng
```

**All the logs are stored in XML format, int the folder `/var/log/-belt/$USERNAME`.** Each file has the following name: `Username_session`, The filename holds the unique ID and which session is stored, so that information is not included in the file.

Code 5 shows our current configuration, we started out with a default file, and then added our code. Where our code starts and end is showed in the file. The important parts are the global options, XML template, CSV template and source statement for TLS communication. The rest is just here for full reference.

To change between XML and CSV format, you have to comment and un-comment the appropriate line in the first log statement for TLS communication.

To make more advanced changes you have to know the basic of how the config file is structured. Syslog-NG uses the same certificate as the webserver, but it can only handle unencrypted certificates, which means that you need to remove the encryption on the certificate, you can do that with the following command: `openssl rsa -in /root/server_cert/key_encrypted.key -out /root/server_cert/key_unencrypted.key`. This example assume that you use RSA and that the key certificates are in the same place as in our configuration.

The behaviour of an incoming log event is based on six configuration:

**Global options** You might want to tweak the global options for performance, see the full Syslog-NG manual<sup>1</sup> for a complete reference of what you can change. Some options should not be set as global, but rather as source specific, you set those in the source statements.

**Source** Here you can say who is allowed to send logs, the format of logs, protocol and tweak some options. You might want to tweak some options, but the rest you probably want to leave as is. You might have to change the path to the certificates.

**Parsers** Sometimes you want to get part of a field, then you can create a parser, as is done to get the session number and event number.

**Templates** This tells how to write the input to file, you have basic if-else statements to determine what to write. Everything in the event can be addressed as a variable. We have XML and CSV output for now. You need to change this if you change what is sent to the server or if you want a different format.

**Destination** This just tells what template to use, and filename.

**Log** This is where you tie it all together. You don't have to change anything here, unless you change something with the above configurations.

We only use TLS for communication, but TCP, and UDP is also possible, but not recommended as this is highly sensitive data. All traffic happens in according to RFC 5424 (Syslog).

Listing 5: syslog-ng.conf file

```

1 @version: 3.3
2 @include "scl.conf"
3
4 # Syslog-ng configuration file, compatible with default Debian
   syslogd
5 # installation.
6
7 # First, set some global options.
8 # We should try to set this for performance
9 options { chain_hostnames(off);
10
11 # This can greatly increase performance
12 # if we use a high value, but we might
13 # loose events in case of a crash
14 # It HAS to be <= log-fetch-limit / max-connections
15 flush_lines(10);
16
17 use_dns(no); use_fqdn(no);
18 owner("root"); group("belt"); perm(0640); stats_freq(0);
19 bad_hostname("^gconfd$"); create_dirs(yes); dir_perm(0750);
20

```

<sup>1</sup><http://www.balabit.com/support/documentation/syslog-ng-ose-3.3-guides/en/syslog-ng-ose-v3.3-guide-admin-en/pdf/syslog-ng-ose-v3.3-guide-admin-en.pdf>

```
21 # Neded to keep unique ID instead of IP address
22 keep_hostname(yes);
23
24 frac_digits(4);
25 };
26
27 #####
28 # Sources
29 #####
30 # This is the default behavior of sysklogd package
31 # Logs may come from unix stream, but not from another machine.
32 #
33 source s_src {
34 system();
35 internal();
36 };
37
38
39 #####
40 ## User defined sources
41 #####
42
43 # Logs from normal TCP connections, we don't use it
44 source s_remote_tcp {
45 tcp(
46 ip(0.0.0.0)
47 port(514)
48 log-fetch-limit(100)
49 );
50 };
51
52 # Logs from UDP, which we don't use
53 source s_remote_udp {
54 udp();
55 };
56
57 # Logs from TLS, which is what we use by default
58 source s_tls_remote_no_auth {
59 tcp(
60 # All IP addresses are accepted, you can use this
61 # to only accept a set of given IP-adresses
62 ip(0.0.0.0)
63
64 # Uses port 1999, 19155 from the outside
65 port(1999)
66
67 # Messages are structured as RFC5424
68 flags(syslog-protocol)
69 flags(validate-utf8)
70
71 # Keep the original timestamps as they are most accurate
72 keep-timestamp(yes)
73
74 tls(
75 key_file("/root/server_cert/key_unencrypted.key")
76 cert_file("/root/server_cert/server_certificate.crt")
77 peer-verify(optional-untrusted)
```

```

78 )
79 # Total number of possible different connections
80 max-connections(1000)
81
82 # Is divided by max-connections, is now 100 for each connection
83 log-iw-size(100000)
84
85 # Maximum number of messages fetched from a single loop, from
    each source
86 log-fetch-limit(100)
87 # encoding(UTF-8)
88 );
89 };
90
91 #####
92 ## User defined parsers
93 #####
94
95 # Splits the msg id into session and an event number
96 parser msgid_segment {
97     csv-parser(columns("MSGID.SESSION", "MSGID.NUM")
98         delimiters("_")
99         flags(escape-none)
100         template("${MSGID}")
101     );
102 };
103
104 # Splits timestamp into ISODATE and milliseconds, with this you can
    easier insert into a database
105 parser time_segment {
106     csv-parser(columns("TIME.DATE", "TIME.MS")
107         delimiters(".")
108         flags(escape-none)
109         template("${ISODATE}")
110     );
111 };
112
113 #####
114 ## User defined templates
115 #####
116
117 # Prints out all data, but does not parse any of it
118 template raw_output {
119     template("\${ISODATE}\",\${MSGID}\",\${PROGRAM}\",\${PID}\",\${SDATA}\
    ",\${MSGONLY}\n");
120 };
121
122 template xml_final {
123 template("${if (\${.SDATA.belt@1.action}\ != \"start\") \"\<
    events><event><num>${MSGID.NUM}</num><date>${ISODATE}</date><type>
    ${.SDATA.belt@1.eventID}</type><action>${.SDATA.belt@1.action}</
    action>${if (\${.SDATA.belt@1.eventID}\ == \"E\") \"\<
    ${.SDATA.belt@1.eventID}\ == \"H\") \"\<relation>${.SDATA.
    belt@1.relation}</relation><flag>${.SDATA.belt@1.flag}</flag>}${
    if (\${.SDATA.belt@1.eventID}\ == \"M\") ${if (\${.SDATA.
    belt@1.action}\ == \"W\") <delta>${.SDATA.belt@1.delta}</delta>
    <X>${.SDATA.belt@1.X}</X><Y>${.SDATA.belt@1.Y}</Y>${if (\${.

```

```

SDATA.belt@1.action}\\" == \"M\") \\" <rectangle><bottomY>${.
SDATA.belt@1.bottY}</bottomY><topY>${.SDATA.belt@1.topY}</topY><
leftX>${.SDATA.belt@1.leftX}</leftX><rightX>${.SDATA.belt@1.
rightX}</rightX></rectangle>)) $(if (\\"${.SDATA.belt@1.eventID}\\"
== \"K\") <value>${.SDATA.belt@1.value}</value><count>${.SDATA.
belt@1.count}</count> $(if (\\"${.SDATA.belt@1.eventID}\\" == \"S\
\") <pid>$PID</pid><program>$PROGRAM</program><elemDescription>${.
SDATA.belt@1.elemDescription}</elemDescription>$(if (\\"${.SDATA.
belt@1.desc}\\" != \\")\") <description>${.SDATA.belt@1.desc}</
description> $(if (\\"${.SDATA.belt@1.value}\\" != \\")\") <value>${.
SDATA.belt@1.value}</value> $(if (\\"${.SDATA.belt@1.bottY}\\" == \
\") \\" \\" <rectangle><bottomY>${.SDATA.belt@1.bottY}</bottomY><
topY>${.SDATA.belt@1.topY}</topY><leftX>${.SDATA.belt@1.leftX}</
leftX><rightX>${.SDATA.belt@1.rightX}</rightX></rectangle>))) $(
if (\\"${.SDATA.belt@1.eventID}\\" == \"H\") $(if (\\"${.SDATA.
belt@1.action}\\" == \"KEY\") <type>${.SDATA.belt@1.type}</type><
language>${.SDATA.belt@1.lang}</language> $(if (\\"${.SDATA.belt@1.
action}\\" == \"RES\") <cpu>${.SDATA.belt@1.cpu}</cpu><memory>${.
SDATA.belt@1.mem}</memory> $(if (\\"${.SDATA.belt@1.action}\\" == \
\"DEV\") <description>${.SDATA.belt@1.desc}</description> $(if (\\"
${.SDATA.belt@1.action}\\" == \"SCR\") <id>${.SDATA.belt@1.num}</
id> $(if (\\"${.SDATA.belt@1.action}\\" == \"SCR_Info\") <id>${.
SDATA.belt@1.num}</id><rectangle><bottomY>${.SDATA.belt@1.bottY
}</bottomY><topY>${.SDATA.belt@1.topY}</topY><leftX>${.SDATA.
belt@1.leftX}</leftX><rightX>${.SDATA.belt@1.rightX}</rightX></
rectangle> $(if (\\"${.SDATA.belt@1.action}\\" == \"HID\") <id>${.
SDATA.belt@1.num}</id>$(if (\\"${.SDATA.belt@1.change}\\" == \
false\") <name>${.SDATA.belt@1.name}</name><type>${.SDATA.belt@1.
type}</type> <change>${.SDATA.belt@1.change}</change> <error/>))
))) <belt/>))))</event>$(if (\\"${.SDATA.belt@1.action}\\" != \
stop\") \\" \\" </events>>\n");
124 };
125
126 # The final CSV format
127 template csv_final {
128 template("${MSGID.NUM},${.SDATA.belt@1.eventID},${.SDATA.belt@1.
action}${if (\\"${.SDATA.belt@1.eventID}\\" == \"M\") $(if (\\"${.
SDATA.belt@1.action}\\" != \"W\") ,${.SDATA.belt@1.X}:${.SDATA.
belt@1.Y} ,${.SDATA.belt@1.delta)} $(if (\\"${.SDATA.belt@1.
eventID}\\" == \"K\") ,${.SDATA.belt@1.value} $(if (\\"${.SDATA.
belt@1.eventID}\\" == \"S\") , $PROGRAM $(if (\\"${.SDATA.belt@1.
eventID}\\" == \"H\") $(if (\\"${.SDATA.belt@1.action}\\" == \"RES\
\" ) ,${.SDATA.belt@1.cpu} ,${.SDATA.belt@1.mem} $(if (\\"${.SDATA.
belt@1.action}\\" == \"HID\") ,${.SDATA.belt@1.num} ,${.SDATA.
belt@1.change}${if (\\"${.SDATA.belt@1.change}\\" == \"true\") \\" \
\" ,${.SDATA.belt@1.name} ,${.SDATA.belt@1.type} ) $(if (\\"${.SDATA.
belt@1.action}\\" == \"SCR_Info\") ,${.SDATA.belt@1.leftX} ,${.
SDATA.belt@1.topY} ,${.SDATA.belt@1.rightX} ,${.SDATA.belt@1.bottY
} ,${.SDATA.belt@1.num} $(if (\\"${.SDATA.belt@1.action}\\" == \"SCR
\") ,${.SDATA.belt@1.num} $(if (\\"${.SDATA.belt@1.action}\\" == \
\"DEV\") ,${.SDATA.belt@1.desc} $(if (\\"${.SDATA.belt@1.action}\\"
== \"KEY\") ,${.SDATA.belt@1.lang} ,${.SDATA.belt@1.type} error)))
))) $(if (\\"${.SDATA.belt@1.eventID}\\" == \"B\") \\" \\" error2))))
,${ISODATE}${if (\\"${.SDATA.belt@1.eventID}\\" == \"H\") \\" \\" $(if
(\\"${.SDATA.belt@1.eventID}\\" == \"B\") \\" \\" ,${.SDATA.belt@1.
relation} ,${.SDATA.belt@1.flag}${if (\\"${.SDATA.belt@1.
elemDescription}\\" == \\" \\" ) \\" \\" ,${.SDATA.belt@1.elemDescription

```



```

    },${PID})$(if (\`${SDATA.belt@1.bottY}\` == \`\`) \`\` ,${SDATA
    .belt@1.leftX},${SDATA.belt@1.topY},${SDATA.belt@1.rightX},${
    SDATA.belt@1.bottY})$(if (\`${SDATA.belt@1.eventID}\` != \`S\`)
    $(if (\`${SDATA.belt@1.count}\` == \`\`) \`\` ,${SDATA.belt@1.
    count}) $(if (\`${SDATA.belt@1.desc}\` == \`\`) \`\` ,${SDATA.
    belt@1.desc})$(if (\`${SDATA.belt@1.value}\` == \`\`) \`\` ,${
    SDATA.belt@1.value}))))\n");
129 };
130
131 #####
132 ## User defined destinations
133 #####
134
135 # Simple destination for performance testing
136 destination d_performance_test {
137     file("/var/log/TEST/performance_test.log");
138 };
139
140 # Simple destination for udp, is not used
141 destination d_udp_remote {
142     file("/var/log/HOSTSUDP/${HOST}");
143 };
144
145 # Simple destination for TCP, is not used
146 destination d_tcp_remote {
147     file ("/var/log/HOSTSTCP/${HOST}");
148 };
149
150 # XML file
151 destination d_xml {
152     file("/var/log/XML/${HOST}_${MSGID.SESSION:-nosession}"
153         template(xml_file)
154     );
155 };
156
157 # Final XML file
158 destination d_xml_final {
159     file("/var/log/belt/${HOST}/${HOST}_${MSGID.SESSION:-nosession}.
160         xml"
161         template(xml_final)
162     );
163 };
164
165 # Final CSV file
166 destination d_csv_final {
167     file("/var/log/belt/${HOST}/${HOST}_${MSGID.SESSION:-nosession}.
168         csv"
169         template(csv_final)
170     );
171
172 # CSV file
173 destination d_csv {
174     file("/var/log/CSV/${HOST}_${MSGID.SESSION:-nosession}"
175         template(csv_file)
176     );

```

```

177 };
178
179 # Plain output
180 destination d_raw {
181     file("/var/log/RAW/${HOST}_${MSGID.SESSION:-nosession}")
182     template(raw_output)
183 };
184 };
185
186 #####
187 ## User defined log statements
188 #####
189
190 # Log description for TLS communication
191 log {
192     source(s_tls_remote_no_auth);
193     parser(msgid_segment); # Need this to parse MSGID
194
195     # Inverse the comment to switch between XML and CSV
196     destination(d_xml_final);
197 # destination(d_csv_final); # CSV output
198
199     # With this we avoid losing messages
200     flags(flow-control);
201 };
202
203 # Log description for UDP communication, is not used
204 log {
205     source(s_remote_udp);
206     destination(d_udp_remote);
207 };
208
209 # Log description for TCP communication, is not used
210 log {
211     source(s_remote_tcp);
212     destination(d_tcp_remote);
213 };
214
215 #####
216 ## End of user defined configuration
217 #####
218
219 # If you wish to get logs from remote machine you should uncomment
220 # this and comment the above source line.
221 #
222 #source s_net { tcp(ip(127.0.0.1) port(1000) authentication(
223     required) encrypt(allow)); };
224 #####
225 # Destinations
226 #####
227 # First some standard logfile
228 #
229 destination d_auth { file("/var/log/auth.log"); };
230 destination d_cron { file("/var/log/cron.log"); };
231 destination d_daemon { file("/var/log/daemon.log"); };
232 destination d_kern { file("/var/log/kern.log"); };

```

```

233 destination d_lpr { file("/var/log/lpr.log"); };
234 destination d_mail { file("/var/log/mail.log"); };
235 destination d_syslog { file("/var/log/syslog"); };
236 destination d_user { file("/var/log/user.log"); };
237 destination d_uucp { file("/var/log/uucp.log"); };
238
239 # This files are the log come from the mail subsystem.
240 #
241 destination d_mailinfo { file("/var/log/mail/mail.info"); };
242 destination d_mailwarn { file("/var/log/mail/mail.warn"); };
243 destination d_mailerr { file("/var/log/mail/mail.err"); };
244
245 # Logging for INN news system
246 #
247 destination d_newscrit { file("/var/log/news/news.crit"); };
248 destination d_newserr { file("/var/log/news/news.err"); };
249 destination d_newsnotice { file("/var/log/news/news.notice"); };
250
251 # Some 'catch-all' logfiles.
252 #
253 destination d_debug { file("/var/log/debug"); };
254 destination d_error { file("/var/log/error"); };
255 destination d_messages { file("/var/log/messages"); };
256
257 # The root's console.
258 #
259 destination d_console { usertty("root"); };
260
261 # Virtual console.
262 #
263 destination d_console_all { file("/dev/tty10"); };
264
265 # The named pipe /dev/xconsole is for the nsole' utility. To use
  it,
266 # you must invoke nsole' with the -file' option:
267 #
268 #     $ xconsole -file /dev/xconsole [...]
269 #
270 destination d_xconsole { pipe("/dev/xconsole"); };
271
272 # Send the messages to an other host
273 #
274 #destination d_net { tcp("127.0.0.1" port(1000) authentication(on)
  encrypt(on) log_fifo_size(1000)); };
275
276 # Debian only
277 destination d_ppp { file("/var/log/ppp.log"); };
278
279 #####
280 # Filters
281 #####
282 # Here's come the filter options. With this rules, we can set which
283 # message go where.
284
285 filter f_dbg { level(debug); };
286 filter f_info { level(info); };
287 filter f_notice { level(notice); };

```

```

288 filter f_warn { level(warn); };
289 filter f_err { level(err); };
290 filter f_crit { level(crit .. emerg); };
291
292 filter f_debug { level(debug) and not facility(auth, authpriv, news
, mail); };
293 filter f_error { level(err .. emerg) ; };
294 filter f_messages { level(info,notice,warn) and
295     not facility(auth,authpriv,cron,daemon,mail,
        news); };
296
297 filter f_auth { facility(auth, authpriv) and not filter(f_debug);
    };
298 filter f_cron { facility(cron) and not filter(f_debug); };
299 filter f_daemon { facility(daemon) and not filter(f_debug); };
300 filter f_kern { facility(kern) and not filter(f_debug); };
301 filter f_lpr { facility(lpr) and not filter(f_debug); };
302 filter f_local { facility(local0, local1, local3, local4, local5,
303     local6, local7) and not filter(f_debug); };
304 filter f_mail { facility(mail) and not filter(f_debug); };
305 filter f_news { facility(news) and not filter(f_debug); };
306 filter f_syslog3 { not facility(auth, authpriv, mail) and not
    filter(f_debug); };
307 filter f_user { facility(user) and not filter(f_debug); };
308 filter f_uucp { facility(uucp) and not filter(f_debug); };
309
310 filter f_cnews { level(notice, err, crit) and facility(news); };
311 filter f_cother { level(debug, info, notice, warn) or facility(
    daemon, mail); };
312
313 filter f_ppp { facility(local2) and not filter(f_debug); };
314 filter f_console { level(warn .. emerg); };
315
316 #####
317 # Log paths
318 #####
319 log { source(s_src); filter(f_auth); destination(d_auth); };
320 log { source(s_src); filter(f_cron); destination(d_cron); };
321 log { source(s_src); filter(f_daemon); destination(d_daemon); };
322 log { source(s_src); filter(f_kern); destination(d_kern); };
323 log { source(s_src); filter(f_lpr); destination(d_lpr); };
324 log { source(s_src); filter(f_syslog3); destination(d_syslog); };
325 log { source(s_src); filter(f_user); destination(d_user); };
326 log { source(s_src); filter(f_uucp); destination(d_uucp); };
327
328 log { source(s_src); filter(f_mail); destination(d_mail); };
329 #log { source(s_src); filter(f_mail); filter(f_info); destination(
    d_mailinfo); };
330 #log { source(s_src); filter(f_mail); filter(f_warn); destination(
    d_mailwarn); };
331 #log { source(s_src); filter(f_mail); filter(f_err); destination(
    d_mailerr); };
332
333 log { source(s_src); filter(f_news); filter(f_crit); destination(
    d_newscrit); };
334 log { source(s_src); filter(f_news); filter(f_err); destination(
    d_newserr); };

```

```

335 log { source(s_src); filter(f_news); filter(f_notice); destination(
      d_newsnotice); };
336 #log { source(s_src); filter(f_cnews); destination(d_console_all);
      };
337 #log { source(s_src); filter(f_cother); destination(d_console_all);
      };
338
339 #log { source(s_src); filter(f_ppp); destination(d_ppp); };
340
341 log { source(s_src); filter(f_debug); destination(d_debug); };
342 log { source(s_src); filter(f_error); destination(d_error); };
343 log { source(s_src); filter(f_messages); destination(d_messages);
      };
344
345 log { source(s_src); filter(f_console); destination(d_console_all);
      destination(d_xconsole); };
346
347 log { source(s_src); filter(f_crit); destination(d_console); };
348
349 # All messages send to a remote site
350 #
351 #log { source(s_src); destination(d_net); };
352
353 ###
354 # Include all config files in /etc/syslog-ng/conf.d/
355 ###
356 @include "/etc/syslog-ng/conf.d/"

```

### 3.3 Syslog protocol

The structure of the Syslog protocol is quite simple, for a more detailed discussion of each field, see the report. Here we will look at all the possible events that can be generated from BeLT.

"TIME" is the full date and time within millisecond accuracy, one example is: "2013-05-03T12:59:37.0766Z", "ID" is the users unique ID, this is a 128-bit value printed as Base 64, without padding. The rest is printed verbatim.

```

<134>1 TIME ID - - 67_1 [belt@1 eventID="B" action="start"]
<134>1 TIME ID - - 67_2 [belt@1 eventID="H" action="KEY" type="7"
  lang="1044"]
<134>1 TIME ID - - 3_3 [belt@1 eventID="H" action="SCR" num="1"
  bottY="768" topY="0" leftX="0" rightX="1366" change="false"]
<134>1 TIME ID - - 67_8 [belt@1 eventID="H" action="SCR" num="1"
  change="true"]
<134>1 TIME ID - - 4_31 [belt@1 eventID="H" action="RES" cpu="2.4696"
  mem="57"]
<134>1 TIME ID - - 4_65 [belt@1 eventID="H" action="DEV" desc="1"]
<134>1 TIME ID - - 67_4 [belt@1 eventID="M" action="M" flag="0"
  X="485" Y="271" relation="0"]
<134>1 TIME ID - - 67_10 [belt@1 eventID="M" action="U" flag="1"
  X="606" Y="422" bottY="546" topY="304" leftX="436" rightX="934"
  relation="6"]

```

```

<134>1 TIME ID - - 67_59 [belt@1 eventID="M" action="W" flag="4"
  delta="-7864320" relation="53"]
<134>1 TIME ID - - 67_55 [belt@1 eventID="K" action="U"
  value="|LWindows|" count="1" relation="54" flag="0"]
<134>1 TIME ID belt_main.exe 1008 67_9 [belt@1 eventID="S" action="FC"
  elemType="4" elemDescription="" relation="6" bottY="546" topY="304"
  leftX="436" rightX="934"]
<134>1 TIME ID explorer.exe |unknown| 4_98 [belt@1 eventID="S"
  action="VC" elemType="32" elemDescription="File Explorer"
  relation="96" value="1"]

```

This list should be somewhat complete of all the categories we can receive of events. There are several variations of each event, but this is the general format. A couple of things worth mentioning is that the order of the elements in the structured data field is irrelevant, the ID of the structured data is always belt@1, every field is case-sensitive and the priority and version value is always 134 and 1 respectively.

### 3.4 Database

To store the data collected from BeLT we have implemented a MySQL database on the server, see the ER-model in figure 1. This database is regularly updated with captured data by a cron job that we have set to run at 00:00am every day. We chose to run it at this time because there will be less users active at this time. Importing the data into the database in a time of little activity will avoid causing performance issues with the server since it will take both time and resources away from processing events from the server. The database is designed to keep the overhead low, but there is still some overhead, especially in the software table.

The database stores the information about each session in the "Session" table where we register the start and end time of each session. Every event is stored in the "Event" table where we store the timestamp, millisecond, type and action about the event. The timestamp is stored as a standard timestamp without millisecond information, instead we store the millisecond information as an integer in a separate field. This is because it makes it easier to search the database based on time since it doesn't have to deal with milliseconds as well as it reduces the possibility of failing if the timestamp format in the database doesn't support millisecond in their timestamp format. It also makes it easier for us to calculate the difference since session start when converting the data to a CSV formatted file.

We have created the backend for a future implementation of "HID" events, but it is removed from the implementation. To implement this feature and the format remains unchanged, add the commented out table to the database along with creating an index for it.

It will then input the captured data specific to the events type and action to the appropriate tables, in the manner shown in the table below.

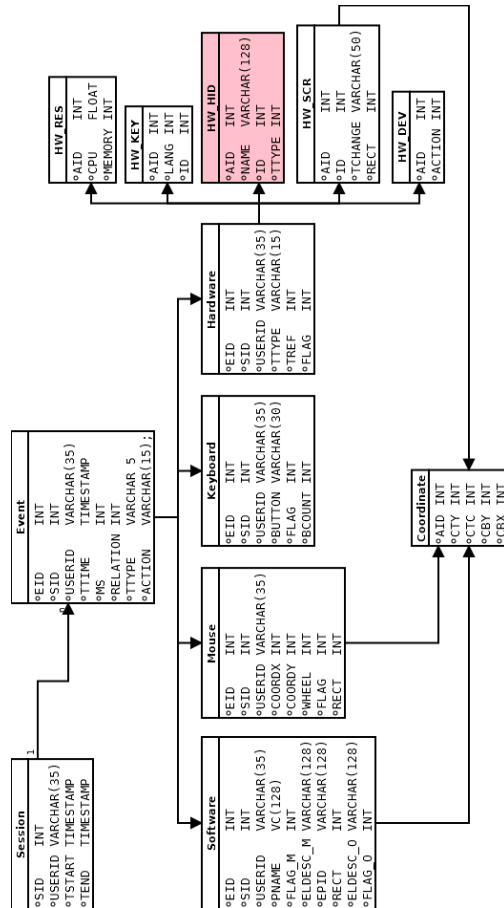


Figure 1: Conceptual model of the database

Overview of table storage			
Type	Action	Main table	Additional tables
M	M/W	Mouse	
M	D/U	Mouse	Coordinates
K	All	Keyboard	
S	All	Software	(*Coordinate)
H	RES	Hardware	HW_RES
H	KEY	Hardware	HW_KEY
H	SCR	Hardware	HW_SCR, (*Coordinate)
H	DEV	Hardware	HW_DEV
<i>Continues on next page...</i>			

Type	Action	Main table	Additional tables
------	--------	------------	-------------------

Table 2: Relations between tables  
(\* )No required

The database is generated with the script in listing 6. This SQL script sets up the database with the indexes we know is needed for the database system. To add more indexes these can be implemented in the bottom of the script along with the rest of the indexes, but must be created with a unique name and after the tables it indexes have been created.

The database script will normally overwrite the entire database by calling the "DROP DATABASE IF EXISTS belt\_syslog;". This is very important to remember to avoid deleting data store in the database. We have left it like this because, if one has to add any indexes or change any fields in the database while it is implemented they should create a completely new script that only performs the desired tasks. This script should only be run when the database should be installed on a fresh system or completely renewed.

Listing 6: "SQL script for generating the database"

```

1 #####
2 # Database generation script
3 #####
4
5 #####
6 # Preamble
7 #####
8 DELIMITER ;
9 DROP DATABASE IF EXISTS belt_syslog;
10 CREATE DATABASE belt_syslog;
11 USE belt_syslog;
12
13 #####
14 # Creating HW-tables
15 #####
16 CREATE TABLE IF NOT EXISTS HW_RES(
17 AID      INT AUTO_INCREMENT PRIMARY KEY,
18 CPU      FLOAT NULL,
19 MEMORY  INT NULL
20 ) ENGINE=MYISAM;
21
22 CREATE TABLE IF NOT EXISTS HW_KEY(
23 AID      INT AUTO_INCREMENT PRIMARY KEY,
24 LANG     INT NULL,
25 ID       INT NULL
26 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
27
28 CREATE TABLE IF NOT EXISTS HW_SCR(
29 AID      INT AUTO_INCREMENT PRIMARY KEY,
30 ID       INT NULL,
31 TCHANGE VARCHAR(50) NULL,
32 RECT     INT NULL
33 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
34

```



```

35 CREATE TABLE IF NOT EXISTS HW_DEV(
36 AID      INT AUTO_INCREMENT PRIMARY KEY,
37 ACTION  INT NULL
38 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
39
40 #-- Table for HID events which was removed from BeLT, but kept if
    needed later
41 #--CREATE TABLE IF NOT EXISTS HW_HID(
42 #--AID      INT AUTO_INCREMENT PRIMARY KEY,
43 #--NAME     VARCHAR(128) NULL DEFAULT NULL,
44 #--ID      INT NULL DEFAULT NULL,
45 #--TTYPE   INT NULL DEFAULT NULL
46 #--) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
47
48
49 #####
50 # Creating Coordinate table for rectangles
51 #####
52 CREATE TABLE IF NOT EXISTS Coordinate(
53 AID      INT AUTO_INCREMENT PRIMARY KEY,
54 CTY     INT NULL,
55 CTX     INT NULL,
56 CBY     INT NULL,
57 CBX     INT NULL
58 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
59
60
61 #####
62 # Creating main event table
63 #####
64 CREATE TABLE IF NOT EXISTS Event(
65 EID      INT NOT NULL,
66 SID      INT NOT NULL,
67 USERID   VARCHAR(35) NOT NULL,
68 TIME     TIMESTAMP NULL DEFAULT NULL,
69 MS       INT NOT NULL DEFAULT 0,
70 RELATION INT NULL DEFAULT NULL,
71 TTYPE    VARCHAR(5) NULL,
72 ACTION   VARCHAR(15) NULL
73 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
74
75
76 #####
77 # Creating main session table
78 #####
79 CREATE TABLE IF NOT EXISTS Session(
80 SID      INT NOT NULL,
81 USERID   VARCHAR(35) NOT NULL,
82 TSTART   TIMESTAMP NULL DEFAULT NULL,
83 TEND     TIMESTAMP NULL DEFAULT NULL
84 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
85
86
87 #####
88 # Creating event type tables
89 #####
90 CREATE TABLE IF NOT EXISTS Software(

```

```

91 EID          INT NOT NULL ,
92 SID          INT NOT NULL ,
93 USERID       VARCHAR(35) NOT NULL ,
94 PNAME        VARCHAR(128) NULL ,
95 FLAG_M       INT          NULL DEFAULT NULL ,
96 ELDESC_M     TEXT          NULL DEFAULT NULL ,
97 EPID         VARCHAR(250) NULL DEFAULT NULL ,
98 RECT         INT          NULL DEFAULT NULL ,
99 ELDESC_O     TEXT          NULL DEFAULT NULL ,
100 FLAG_O       INT          NULL DEFAULT NULL
101 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
102
103 CREATE TABLE IF NOT EXISTS Mouse(
104 EID          INT NOT NULL ,
105 SID          INT NOT NULL ,
106 USERID       VARCHAR(35) NOT NULL ,
107 COORDX      INT NULL DEFAULT 0,
108 COORDY      INT NULL DEFAULT 0,
109 WHEEL       INT NULL DEFAULT NULL ,
110 FLAG        INT NULL DEFAULT NULL ,
111 RECT        INT NULL DEFAULT NULL
112 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
113
114 CREATE TABLE IF NOT EXISTS Keyboard(
115 EID          INT NOT NULL ,
116 SID          INT NOT NULL ,
117 USERID       VARCHAR(35) NOT NULL ,
118 BUTTON       VARCHAR(30) NULL ,
119 FLAG        INT          NULL ,
120 BCOUNT     INT          NULL DEFAULT NULL
121 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
122
123 CREATE TABLE IF NOT EXISTS Hardware(
124 EID          INT NOT NULL ,
125 SID          INT NOT NULL ,
126 USERID       VARCHAR(35) NOT NULL ,
127 TREF        INT          NULL ,
128 FLAG        VARCHAR(10) NULL
129 ) ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
130
131
132 #--#####
133 #--#   Index generation
134 #--#####
135 ALTER TABLE `Session` ADD UNIQUE INDEX (SID, USERID);
136 ALTER TABLE `Session` ADD INDEX (SID);
137 ALTER TABLE `Session` ADD INDEX (USERID);
138
139 ALTER TABLE Event ADD UNIQUE INDEX (EID, SID, USERID);
140 ALTER TABLE Event ADD INDEX (EID);
141 ALTER TABLE Event ADD INDEX (SID);
142 ALTER TABLE Event ADD INDEX (USERID);
143
144 ALTER TABLE Hardware ADD UNIQUE INDEX (EID, SID, USERID);
145 ALTER TABLE Hardware ADD INDEX (EID);
146 ALTER TABLE Hardware ADD INDEX (SID);
147 ALTER TABLE Hardware ADD INDEX (USERID);

```

```

148
149 ALTER TABLE Software ADD UNIQUE INDEX (EID, SID, USERID);
150 ALTER TABLE Software ADD INDEX (EID);
151 ALTER TABLE Software ADD INDEX (SID);
152 ALTER TABLE Software ADD INDEX (USERID);
153
154 ALTER TABLE Mouse ADD UNIQUE INDEX (EID, SID, USERID);
155 ALTER TABLE Mouse ADD INDEX (EID);
156 ALTER TABLE Mouse ADD INDEX (SID);
157 ALTER TABLE Mouse ADD INDEX (USERID);
158
159 ALTER TABLE Keyboard ADD UNIQUE INDEX (EID, SID, USERID);
160 ALTER TABLE Keyboard ADD INDEX (EID);
161 ALTER TABLE Keyboard ADD INDEX (SID);
162 ALTER TABLE Keyboard ADD INDEX (USERID);
163
164 ALTER TABLE Coordinate ADD UNIQUE INDEX (AID);
165 ALTER TABLE HW_RES ADD UNIQUE INDEX (AID);
166 ALTER TABLE HW_KEY ADD UNIQUE INDEX (AID);
167 ALTER TABLE HW_SCR ADD UNIQUE INDEX (AID);
168 ALTER TABLE HW_DEV ADD UNIQUE INDEX (AID);
169 #--ALTER TABLE HW_HID ADD UNIQUE INDEX (AID);

```

## Database Import script

The import script(7) is scheduled to run to run by Cron. The script retrieves the files stored in the "/var/log/belt\_archive" directory, retrieves and inserts the data into the database to the correct fields. This script is dependant on the file format of the XML and the database layout. Any changes to either the XML or database has to be implemented in the import script and tested before letting it run on the actual server.

In addition it can also perform the task of converting an XML formatted file into an "UTF-8" formatted file. Which allows for use of special symbols like "©" without corrupting the file format.

This script performs its tasks automatically without any user interaction and outputs all errors into a log file inside the base directory "/var/log/belt/". The log file will contain information about each test it ran and how long it spent parsing all the files. All log events are formatted using a common tag before the description follows on the same line.

The script will first retrieve all XML files in the directory "/var/log/belt" and create a list with their absolute pathnames. The script will then start by running the "tagRemoval" function which creates a UTF-8 encoded file with all unwanted symbols replace by a HTML escaped symbol text. When this is finished we check if the file is well formed by trying to parse it. If it raises an exception we know it is either corrupt or not finished. if it passes the test we continue by retrieving all "<event>" elements and go through them one by one. This is done systematically based on what event type it is. If an element is empty we'll get the text "None" which we have to take into accordance when parsing them. If this happens we have to insert a "NULL" to the database

instead. This all happens in the "getVal" and "wrap" function which creates the correct text and format for the database.

After parsing all the files we move the original file, without HTML escaped characters to the archive "/var/log/belt\_archive". This way we avoid adding the same information more than twice thus removing the data which would otherwise corrupt our dataset.

To change the connection information of the MySQL connection object change the information in the following line inside the header declaration for variables and objects.

```
1 database = MySQLdb.connect(host="localhost", user="root", passwd="
  toor", db="belt_syslog")
```

To change the different paths we are using change the following lines in the header: "pBase" is the base directory where the logs are being stored by Syslog-NG. "tmpFile" is our temporary file which we are storing the files temporary while converting and parsing them. The "logFile" is the path and filename where we store our log file. Here we add a timestamp to allow for multiple files to be created inside the same directory. The "archive" variable holds the absolute path of our archive folder, where we store our parsed XML files.

```
1 pBase = "/var/log/belt/"
2 tmpFile = pBase+"tmp.xml"
3 logFile = pBase+"BeLT_import_"+logTime.strftime("%Y%m%d_%H%M%S") +
  ".log"
4 archive = "/var/log/belt_archive/"
```

To change what encoding we are converting from an to change the following two lines in the header. "sourceEncoding" is the format we are reading the first time, and "targetEncoding" is the format we are writing the file to.

```
1 sourceEncoding = "iso -8859-1"
2 targetEncoding = "utf-8"
```

Listing 7: "Python script to import data to the database"

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # Author: Magnus Øverbø - 05.05.2013-13.05.2013
4 # To parse from XML to XML run the script with the XML argument
5 # Which will cause this script to store the parsed xml as
6 # UTF-8 encoded and html escaped XML code
7
8 #####
9 # Import libraries
10 #####
11 import xml.etree.ElementTree as ET # For main xml process
12 import MySQLdb # For MySQL db
13 import os # For listing/delete
14 import cgi # Parsing unwanted tags
15 import sys # arguments
16 from xml.parsers.expat import ParserCreate, ExpatError, errors #
  For well-formed check
17 import codecs
18 import datetime
```

```

19
20 #####
21 # Create a variables/objects
22 #####
23 database = MySQLdb.connect(host="localhost", user="root", passwd="
    toor",db="belt_syslog")
24 query = database.cursor()
25 count = 0
26 totCount = 0
27 g_wf = 0
28 logTime = datetime.datetime.now()
29 pBase = "/var/log/belt/"
30 tmpFile = pBase+"tmp.xml"
31 logFile = pBase+"BeLT_import_"+logTime.strftime("%Y%m%d_%H%M%S")+
    ".log"
32 archive = "/var/log/belt_archive/"
33 sourceEncoding = "iso-8859-1"
34 targetEncoding = "utf-8"
35 x2x = False
36 if len(sys.argv) >= 2:
37     if sys.argv[1] == "XML":
38         x2x = True
39
40 #####
41 # Grab all xml files from dirs
42 #####
43 pFileArr = []
44 for pUsers in os.listdir(pBase): #Find user folders
45     if os.path.isdir(pBase+pUsers): #If it's a folder
46         for pFile in os.listdir(pBase+pUsers):#Find items inside the
            folders
47             if not os.path.isdir(pBase+pUsers+"/"+pFile): # Grab only the
                files
48                 pFileArr.append(pBase+pUsers+"/"+pFile) # Append to
                    list
49
50 #####
51 # Functions
52 #####
53 ## Goes through a string and replaces any special chars
54 def removeTag(s):
55     r=""
56     for c in s:
57         r += cgi.escape(c) #replace special char with encoded symbol
58     return r
59
60 ## Walk through the file, grab special elements
61 ## remove special characters and write it to a tmp file
62 def tagRemoval(fname):
63     #Fields to check for invalid fields in
64     lst = ["value", "pid", "program", "description", "elemDescription"
        ]
65     f = open(fname, "r") # existing file
66     b = open(tmpFile+".utf8.xml", "w") #converted file to UTF8
67     b.write(unicode(f.read(), sourceEncoding).encode(targetEncoding))
68     b.close()
69     f.close()

```

```

70 b = open(tmpFile+".utf8.xml", "r") # Converted file to UTF8
71 a = open(tmpFile, "w")           # Temporary file
72 for l in b:                       # For each line in file
73     for i in lst:                 # For each element to check
74         if l.find("<"+i+">") > 0: # If element in list
75             n = l[:l.find("<"+i+">")+2+len(i)] # grab text up to
              element
76             n = n + removeTag(l[l.find("<"+i+">")+2+len(i):l.find("</"+
              i+">")])
77                                     # remove special char
              from text
78             n = n + l[l.find("</"+i+">"):l.find("</"+
              i+">")+2+len(i)] # add the rest of the
              line back
79             l = n # set the read line like the new one
80         a.write(l)
81     a.close()
82     b.close()
83     try:
84         os.remove(tmpFile+".utf8.xml")
85     except Exception, e:
86         print tmpFile+".utf8.xml couldn't be removed"
87
88 ## Checks if the the XML file is well formed
89 def isWF(fname):
90     parser = ParserCreate()
91     parser.ParseFile(open(fname, "r"))
92
93 ## Insert queries takes the table name, and a list with fields
94 # and list with values to insert
95 def dbSend(table, field, value):
96     stmt = "INSERT INTO "+table+"("+field+") VALUES("+value+");"
97     try:
98         query.execute(stmt)
99     except Exception, e:
100         log.write("SQL_ERROR: " + str(e)+"\n")
101         log.write("SQL_STMT: "+stmt+"\n")
102         print "SQL ERROR: "+ stmt;
103
104 ## Insert a custom querie
105 def dbSpecial(stmt):
106     ret = ""
107     try:
108         query.execute(stmt)
109         ret = query.lastrowid
110     except Exception, e:
111         log.write("SQL_ERROR: " + str(e)+"\n")
112         log.write("SQL_STMT: "+stmt+"\n")
113         ret = None;
114     return str(ret)
115
116 ## Grabs the value with error checking,
117 ## returns the value or NULL
118 def getVal(el, tag):
119     out = ""
120     tmp = el.find(tag) #Grab element from current event
121     if tmp is None:    #If tmp is none(empty or nonexistant)
122         out = "NULL"  # Set it to NULL

```

```

123     else:                                #If it exists
124         out = unicode(tmp.text)# grab the elements node value as a
            string
125         if out == "None": # if it contained nothing "None" is received
126             out = "NULL" # if so set it to NULL
127         return out.encode('utf-8') # Return the retrieved value
128
129     ## Wraps the string value to avoid problems with space an NULL
130     def wrap(s):
131         if str(s) == 'NULL':
132             s = ",NULL"
133         else:
134             s = ","+str(s)+","
135         return s
136
137     ## Creates an entry in the Coordinate table and returns its ID
138     def newRect(e):
139         ty = getVal(event, "rectangle/topY")
140         tx = getVal(event, "rectangle/leftX") # Grab values
141         by = getVal(event, "rectangle/bottomY")
142         bx = getVal(event, "rectangle/rightX")
143         s = "INSERT INTO Coordinate (CTY,CBY,CTX,CBX) VALUES('"
144         t = dbSpecial(s + ty + "','" + wrap(by) + wrap(tx) + wrap(bx) +")";
145         return t
146
147     #####
148     # Files loop
149     #####
150     log = open(logFile, "w+")
151     log.write("#####\n"
152 )
152     log.write("Full Start: "+str(datetime.datetime.now())+"\n")
153     log.write("#####\n"
154 )
154     for bfile in pFileArr:
155         log.write("FILE: "+bfile+"\n")
156         bSID = bfile[bfile.rfind("_")+1:bfile.rfind(".")]
157         bUID = bfile[bfile.rfind("/")+1:bfile.rfind("_")]
158         log.write("PARSING_START: "+str(datetime.datetime.now())+"\n")
159         tagRemoval(bfile) #Escape symbols
160         try: #If well formed set global g_wf to 1
161             isWF(tmpFile)
162             g_wf = 1
163             if x2x == True:
164                 os.rename(tmpFile, archive+bUID+"_"+bSID+".utf8.xml")
165                 log.write("FILE: Archived as UTF-8 encoded XML\n")
166                 #os.remove(bfile)
167             else:
168                 #os.rename(bfile, archive+bUID+"_"+bSID+".xml")
169                 log.write("FILE: Archived as ISO encoded XML\n")
170         except Exception, e: #If exception and not well formed set g_wf
171             to 0
172             g_wf = 0 #Disable importing this file
173             os.remove(tmpFile) #Remove tmpfile because the session is
174             ongoing or corrupt
175             log.write("FILE_ERROR: Session is corrupted or an ongoing

```

```

    session\n")
174
175 if g_wf == 1 and x2x == False: ## if XML is well formed
176     count = 0
177     xmltree= ET.parse(tmpFile)    # Load the well formed file into
    xmltree
178     root    = xmltree.getroot()    # Grab the root element <
    events>
179     events = root.findall("event") # Grab all <event> elements
    into a list
180
181     #####
182     # Event loop
183     #####
184     for event in events:           # Iterate over the event
    list
185         count += 1                # counts number of
    elements parsed
186         bEvent = getVal(event, "num[1]") # Fetch event counter
187         bType  = getVal(event, "type[1]") # Fetch type as text
188         bAction = getVal(event, "action[1]") # Fetch action as text
189         bFlag   = getVal(event, "flag[1]") # Fetch action as text
190         bRel    = getVal(event, "relation") # Fetch action as text
191         bTime   = getVal(event, "date")    # Fetch time as text
192         bMS     = bTime[bTime.find('.')+1:bTime.find('+')] #grab
    milliseconds
193         bTime   = bTime[:bTime.find('T')] + " " + bTime[bTime.find('T')
    +1:bTime.find('.')]
194         field   = "USERID, SID, EID, TTYPE, ACTION, TTIME, MS,
    RELATION"
195         value   = ""+bUID+"'+wrap(bSID)+wrap(bEvent) + wrap(bType) +
    wrap(bAction) + wrap(bTime) + wrap(bMS) + wrap(bRel)
196         dbSend('Event', field, value)     #Insert event to databse
197
198     #####
199     # BeLT system event
200     #####
201     if bType == "B":
202         if bAction == "start": ## if Start event
203             value = "" + bUID + "' + wrap(bSID) + wrap(bTime)
204             dbSend('Session', 'USERID, SID, TSTART', value)
205         elif bAction == "stop":## if Stop event update row in table
206             stmt = "UPDATE Session SET TEND='"+bTime+"' WHERE USERID='
    "+bUID+"' AND SID='"+bSID+"';"
207             dbSpecial(stmt)
208
209     #####
210     # Keyboard event
211     #####
212     elif bType == "K":
213         v1 =getVal(event, "value")
214         v2 =getVal(event, "count")
215         field = "USERID, SID, EID, BUTTON, FLAG, BCOUNT"
216         value = ""+bUID+"'+wrap(bSID)+wrap(bEvent)+wrap(v1)+wrap(
    bFlag)+wrap(v2)
217         dbSend('Keyboard', field, value)
218

```



```

219 #####
220 # Mouse event
221 #####
222 elif bType == 'M':
223     field = "USERID, SID, EID, FLAG"
224     value = "'" + bUID + "'" + wrap(bSID) + wrap(bEvent) + wrap(bFlag)
225     ## Press/Release of button
226     if bAction == "D" or bAction == "U":
227         tid = newRect(event)
228         v1 = getVal(event, "X")
229         v2 = getVal(event, "Y")
230         field = field + ", RECT, COORDX, COORDY"
231         value = value + wrap(tid) + wrap(v1) + wrap(v2)
232     ## Movement
233     elif bAction == "M":
234         v1 = getVal(event, "X")
235         v2 = getVal(event, "Y")
236         field = field + ", COORDX, COORDY"
237         value = value + wrap(v1) + wrap(v2)
238     ## Wheel action
239     elif bAction == "W":
240         v1 = getVal(event, "delta")
241         field = field + ", WHEEL"
242         value = value + wrap(v1)
243         dbSend('Mouse', field, value)
244
245 #####
246 # Hardware information
247 #####
248 elif bType == 'H':
249     ## If Keyboard information
250     if bAction == "KEY":
251         v1 = getVal(event, "language")
252         v2 = getVal(event, "type[2]")
253         s="INSERT INTO HW_KEY(LANG, ID) VALUES('"+v1+"'" + wrap(v2) + "
254         );"
255     ## If Human interface Device information
256     ## NOT IN USE
257     elif bAction == "HID":
258         v1 = getVal(event, "name")
259         v2 = getVal(event, "id")
260         v3 = getVal(event, "type[2]")
261         s="INSERT INTO HW_HID(NAME, ID, TTYPE) VALUES('"+v1+"'" + wrap
262         (v2) + wrap(v3) + ");"
263
264     ## if Screen information
265     elif bAction == "SCR_Info":
266         if getVal(event, "rectangle/leftX") != "NULL":
267             tid = newRect(event) #add the coordinates and return
268             index
269             s="INSERT INTO HW_SCR(ID, RECT) VALUES('"+getVal(event, "
270             id") + "'" + wrap(tid) + ");"
271
272     ## if Screen change
273     elif bAction == "SCR":
274         s="INSERT INTO HW_SCR(ID) VALUES('"+getVal(event, "id") + "
275         ');";

```

```

271
272     ## If resource information
273     elif bAction == "RES":
274         v1 = getVal(event, "cpu")
275         v2 = getVal(event, "memory")
276         s="INSERT INTO HW_RES(CPU, MEMORY) VALUES(' +v1+''+wrap(
           v2)+'");"
277     ## if Device information
278     elif bAction == "DEV":
279         v1 = getVal(event, "description")
280         s="INSERT INTO HW_DEV(action) VALUES(' + v1 +'');"
281
282     tid = dbSpecial(s) ## get index for Hardware.TREF
283
284     ## Insert information and the reference to the Hardware
           table
285     field = "USERID, SID, EID, TREF, FLAG"
286     value = "'"+bUID+"'"+wrap(bSID)+wrap(bEvent)+wrap(tid)+wrap(
           bFlag)
287     dbSend('Hardware', field, value)
288
289     #####
290     # Software information
291     #####
292     elif bType == 'S':
293         field = "USERID, SID, EID, PNAME, FLAG_M, ELDESC_M, EPID"
294         value = "' + bUID + "' + wrap(bSID)+wrap(bEvent)+wrap(
           getVal(event, "program"))+wrap(bFlag)
295         value = value+wrap(getVal(event, "elemDescription"))+wrap(
           getVal(event, "pid"))
296     ## Text Change
297     if bAction == "TC":
298         field = field + ", ELDESC_0"
299         value = value + wrap(getVal(event, "description"))
300     ## Visual Change
301     elif bAction == "VC":
302         field = field + ", FLAG_0"
303         value = value + wrap(getVal(event, "value"))
304     ## if it is a rectangle present
305     if getVal(event, "rectangle/leftX") != "NULL":
306         tid = newRect(event)
307         field = field + ", RECT"
308         value = value + wrap(tid)
309     dbSend("Software", field, value)
310
311     tid = "NULL"
312     field = "NULL"
313     value = "NULL"
314     v1 = "NULL"
315     v2 = "NULL"
316     v3 = "NULL"
317     log.write("PARSING_END: "+str(datetime.datetime.now())+"\n")
318     log.write("STATISTIC: " +str(count)+" events was parsed\n")
319     totCount= totCount + count
320     ##End well formed check
321     ##End file iteration
322     log.write("#####\n")

```

```

)
323 log.write("FULL END: "+str(datetime.datetime.now())+"\n")
324 log.write("TIME SPENT: "+str(datetime.datetime.now()-logTime)+"\n")
325 log.write("EVENTS PARSED: "+ str(totCount) +"\n")
326 log.write("#####\n")
)
327 log.close()

```

### 3.5 Export XML to XML

The import script also has a second function which is to parse an existing XML file and store it encoded in "UTF-8". This done by passing the argument "XML" to the script in the following manner `./import.py XML` this will cause the importscript to skip the task of inserting data to the database and instead move the temporary XML file to the archive with the name `"[UID]_[SID].utf8.xml"`.

What happens in this run is the following, the script will retrieve the all files in the folders, then it will convert them to "UTF-8" format and store them in a temporary location one by one. This file is then read by the script and for each field we know could contain invalid characters we take the text string from and parse it char by char and escaping any invalid signs. I.e the "&" sign will be encoded to "&amp;". This final file is then stored as a "UTF-8" formatted file which is well formatted and contains Unicode characters.

### 3.6 Export database to CSV

This script(8) is controlled either by a command line interface or an argument. By specifying the argument "GETALL" one will automatically extract all sessions from all users in the database and generate CSV formatted files. We have created this script in Python to make it run as fast as possible. It will retrieve data using the specified user and password to the database, to change user/password/database this has to be changed in the start of the script when configuring the database connection.

Most of the configurations for this script lies in its header under "Create a variables/objects". Here we have the connection to our database, our path variables, required information and handling of the argument one can provide.

Otherwise to change any data gathering one must change the variables according to the database. Python's library for MySQL connection does not provide a functionality for retrieving the fields based on names when returning the fetched values. Because of this we have created a small wrapper that creates a list with a dictionary containing the mapping of the fields names and their value for each row returned by the query. To change the connection information of the MySQL connection object change the information in the following line

```

1 database = MySQLdb.connect(host="localhost", user="root", passwd="
toor", db="belt_syslog")

```

For each session it will create a file in the directory specified in the header, where it will store the CSV formatted files with the filename `"[UID]_[SID].csv"`.

To change this, one must to edit the line at the bottom where one specifies the filename format.

```
1 CSV_FILE = open(dirSave+str(uid)+"_"+str(sid)+".csv", "w")
```

To change the storage folder one has to change the following line in the header which declares the full path to the base path directory.

```
1 dirSave = "/home/belt/belt_csv/"
```

To change the name and directory of the log file change the following line in the header:

```
1 logFile = dirSave + "BeLT_CSV_" + str(datetime.strftime(datetime.now(), "%Y%m%d_%H%M%S")) + ".log"
```

To change the default value for an empty field. Which occurs when the field in the database is empty(NULL), change the EMPTY variable in the header

```
1 EMPTY = "|empty|"
```

Listing 8: "Python script to generate CSV files from data in the database"

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # Magnus Øverbø - 05.05.2013-12.05.2013
4
5 #####
6 # Import python libraries
7 #####
8 import MySQLdb      # MySQL connection
9 from datetime import datetime # timestamp conversion
10 import cgi          # escape/unescape html formatted text
11 import os           # Dir list
12 import sys          #
13
14 #####
15 # Create variables/objects
16 #####
17 db = MySQLdb.connect(host="localhost", user="root", passwd="
18     toor", db="belt_syslog")
19 query = db.cursor()
20 dirSave = "/home/belt/belt_csv/"
21 logFile = dirSave+"BeLT_CSV_"+str(datetime.strftime(datetime.now(),
22     "%Y%m%d_%H%M%S"))+".log"
23 retMap = {}
24
25 EX_EXTRA= ["H", "B"]
26 CSV_LIST = []
27 TIME_T = None
28 TIME_M = int
29 EMPTY = "|empty|"
30 GETALL = False
31 if len(sys.argv) >= 2:
32     if sys.argv[1] == "GETALL":
33         GETALL = True;
34
35 #####
```

```

34 # Functions
35 #####
36 ## Retrive sessions from a UID
37 def getDict(uid, check):
38     global retMap
39     os.system("clear")
40     sids = []
41     query.execute("SELECT DISTINCT SID FROM Session WHERE USERID='"+
uid+"' ORDER BY SID ASC;")
42     res=query.fetchall()
43     if check: ## IF GRAB EVERYTHING
44         for i in range(0,len(res)):
45             sids += [int(res[i][0])]
46         retMap.update({uid:sids})
47     else:
48         print "\t#####"
49         print "\t " + uid      ##Menue for selecting sesions from user
50         print "\t#####"
51         for i in range(0,len(res)):
52             print "\t" + str(res[i][0]),
53             if i%5 == 4:
54                 print ""
55             print "\t(A)ll\t- Select all from user"
56             print "\t#####"
57             ans = raw_input("\t Your choices: ")
58             ans = ans.split()
59             if 'A' in ans:      ##IF A grab all
60                 for i in range(0,len(res)):
61                     sids += [int(res[i][0])]
62                 retMap.update({uid:sids})
63             else:
64                 retMap.update({uid:ans}) #add list of sids to the map
65
66 ## Retrieve UIDs and sessions from DB
67 def getMapping():
68     os.system("clear")
69     uids = []
70     query.execute("SELECT DISTINCT USERID FROM Session ORDER BY
USERID ASC;")
71     res = query.fetchall()
72     for i in res:      ##ad uids to list
73         uids += [i[0]]
74
75     if not GETALL:    # if get all, retrieve all UIDs automatically
76         print "\t SELECT USER IDS FROM LIST"
77         print "\t#####"
78         for i in range(0, len(uids)):      ##Print all uids
79             print "\t " + str(i) + "\t- " + str(uids[i])
80         print "\t(A)ll \t- Select sessions manually from every user"
81         print "\t(G)rab\t- Grab ALL sessions from EVERY user"
82         print "\t#####"
83         ans = raw_input("\t Your choices: ")
84         ans = ans.split()      ##Split sids into a list
85
86         if "G" in ans:      #Grabs all UIDS and SIDS
87             for i in uids:
88                 getDict(i, True)      ##Fetch all sids from all uid

```

```

89     elif "A" in ans: #If select all uids
90         for i in uids:
91             getDict(i, False) ##Fetch sids manually from all uidk
92         else:
93             for i in range(0, len(ans)):
94                 getDict(uids[int(ans[i])], False) ##Choose sids for UID
95     else: #IF GETALL IS TRUE grab all UIDS
96         for i in uids:
97             getDict(i, True) ##Fetch all sids from all uid
98
99 ##Fetch data as a dictionary
100 def fetch(field, table, clause):
101     arr = []
102     stmt = "SELECT "+field+" FROM "+table+" WHERE "+clause+";"
103     field = field.replace(" ", "")#creates a list from fields
104     field = field.split(",") #
105     query.execute(stmt) #queries the server
106     res = query.fetchall() #grab all returned values
107     for i in res: #For all rows returned
108         dic = {} #
109         for j in range(0, len(i)):#correlate fields and values to dict
110             dic.update({ field[j] : str(i[j]) })
111         arr.append(dic) ##Append dict to list
112     return arr ##return array with dictionary of FIELD:VALUE
113
114 ## Get rectangle
115 def getRect(aid):
116     rect = fetch("CTY, CTX, CBY, CBX", "Coordinate", "AID='"+str(aid)
117     +"'")[0]
118     CV(rect['CTX']) #top-left
119     CV(rect['CTY']) #top-top
120     CV(rect['CBX']) #bottom-right
121     CV(rect['CBY']) #bottom-bottom
122
123 #Retrieves the value field between ACTION and TIME
124 def getValue(row, uid, sid):
125     global CSV_LIST
126     clause = "USERID='"+str(uid)+"' AND SID='"+str(sid)+"' AND EID='"+
127     +str(row['EID'])+"'"
128     if row['TTYTYPE'] == 'K': #KEYBOARD EVENT
129         x = fetch('BUTTON', 'Keyboard', clause)[0]
130         CV(x['BUTTON'], True)
131
132     elif row['TTYTYPE'] == 'M' and row['TTYTYPE'] == 'W':
133         x = fetch('WHEEL', 'Mouse', clause)[0]
134         CV(x['WHEEL'])
135
136     elif row['TTYTYPE'] == 'M':
137         x = fetch('COORDX, COORDY', 'Mouse', clause)[0]
138         CV(x['COORDX']+" "+x['COORDY'])
139
140     elif row['TTYTYPE'] == 'S':
141         x = fetch('PNAME', 'Software', clause)[0]
142         CV(x['PNAME'], True)
143
144 #Not in use, but left in to be enabled when HID is used
145     elif row['TTYTYPE']=='H' and row['ACTION']=='HID':

```

```

144     aid = fetch("TREF, FLAG", "Hardware", clause)[0]
145     x = fetch('NAME, ID, TTYPE', 'HW_HID', "AID='"+str(aid['TREF'])
146           +"'")[0]
147     if x['NAME'] != 'None':
148         CV(x['NAME'])
149         CV(x['ID'])
150         CV(x['TTYPE'])
151
152 elif row['TTYPE']=='H' and row['ACTION']=='KEY':
153     aid = fetch("TREF, FLAG", "Hardware", clause)[0]
154     x = fetch("ID, LANG", "HW_KEY", "AID='"+str(aid['TREF'])+"'"
155           "[0]
156           CV(x['LANG'])
157           CV(x['ID'])
158
159 elif row['TTYPE']=='H' and row['ACTION']=='RES':
160     aid = fetch("TREF, FLAG", "Hardware", clause)[0]
161     x = fetch("CPU, MEMORY", "HW_RES", "AID='"+str(aid['TREF'])+"'"
162           "[0]
163           CV(x['CPU'])
164           CV(x['MEMORY'])
165
166 elif row['TTYPE']=='H' and row['ACTION']=='SCR':
167     aid = fetch("TREF, FLAG", "Hardware", clause)[0]
168     x = fetch("RECT, ID", "HW_SCR", "AID='"+str(aid['TREF'])+"'"
169           "[0]
170           if x['RECT'] != 'None':
171               getRect(x['RECT'])
172               CSV_LIST[2]='SCR_Info'
173           CV(x['ID'])
174
175 elif row['TTYPE']=='H' and row['ACTION']=='DEV':
176     aid = fetch("TREF, FLAG", "Hardware", clause)[0]
177     x = fetch("ACTION", "HW_DEV", "AID='"+str(aid['TREF'])+"'"
178           "[0]
179           CV(x['ACTION'])
180
181 ## Get end fields from flag and out
182 def getExtra(row, uid, sid):
183     clause = "USERID='"+str(uid)+"' AND SID='"+str(sid)+"' AND EID='"+
184           +str(row['EID'])+"'"
185     if row['TTYPE'] == 'K':
186         x = fetch("FLAG, BCOUNT", "Keyboard", clause)[0]
187         CV(x['FLAG'])
188         if row['ACTION']=='U' and int(x['BCOUNT']) >= 2:
189             CV(x['BCOUNT'])
190
191 elif row['TTYPE'] == 'M':
192     x = fetch("FLAG, RECT", "Mouse", clause)[0]
193     CV(x['FLAG'])
194     if row['ACTION'] in ['U', 'D']:
195         if x['RECT'] != 'None':
196             getRect(x['RECT'])
197
198 elif row['TTYPE'] == 'S':
199     x=fetch("FLAG_M, ELDESC_M, EPID, RECT, ELDESC_0, FLAG_0", "Software"
200           , clause)[0]
201     CV(x['FLAG_M'])

```

```

195     CV(x['ELDESC_M'], True)
196     CV(x['EPID'], True)
197     if row['ACTION'] == "TC":
198         CV(x['ELDESC_O'], True)
199     if row['ACTION'] == "VC":
200         CV(x['FLAG_O'])
201     if x['RECT'] != 'None':
202         getRect(x['RECT'])
203
204 #SET FIELD TO EMPTY
205 def NA(CAP=False):    #creates an empty value in the list
206     global CSV_LIST
207     if not CAP:
208         CSV_LIST += [EMPTY]
209     else:
210         CSV_LIST += [''+EMPTY+'']
211
212 ## Returns the value of the field or N/A
213 def CV(field, CAP=False):
214     global CSV_LIST
215     if field == 'None':
216         NA(CAP)
217     else:
218         if not CAP:
219             CSV_LIST += [str(field)]
220         else:
221             CSV_LIST += [''+str(field)+'']
222
223 #WRITE CSV FORMAT
224 def WRITE():    ##creates a CSV string
225     global CSV_LIST
226     count = 0
227     CSVString = ""
228     EOL = len(CSV_LIST)
229     for val in CSV_LIST:
230         count += 1
231         if count < EOL:
232             CSVString += val + ","
233         else:
234             CSVString += val
235     return CSVString+"\n"
236
237 ##GENERATE CORRECT TIMESTAMP VALUE
238 def timestamp(T, M, A):
239     global CSV_LIST
240     global TIME_T
241     global TIME_M
242     if str(A) == "start":
243         TIME_T = datetime.strptime(T, "%Y-%m-%d %H:%M:%S")
244         TIME_M = int(M)
245         if TIME_M < 10:
246             M = ".000"+M
247         elif TIME_M < 100:
248             M = ".00"+M
249         elif TIME_M < 1000:
250             M = ".0"+M
251     else:

```



```

252     M = "."+M
253     CV(str(T+M))
254     else:
255         if T is None or M is None:
256             NA()
257         else:
258             dt = (datetime.strptime(T, "%Y-%m-%d %H:%M:%S") - TIME_T).
                total_seconds()
259             CV(int((dt*10000)+(int(M)-TIME_M)))
260
261 #####
262 # Database loop
263 #####
264 LOG = open(logFile, "w")
265 TSTART = datetime.now()
266 LOG.write("TIME_START_ALL: "+str(TSTART)+"\n")
267 getMapping()
268 for uid in retMap.keys():
269     for sid in retMap[uid]:
270         ## Fetch all events from Event table
271         TSTARTFILE = datetime.now()
272         LOG.write("TIME_START: "+str(TSTARTFILE)+" "+str(uid)+" "+str(
                sid)+"\n")
273         table = "Event"
274         field = "SID, EID, TTIME, MS, RELATION, TTYPE, ACTION"
275         clause = "USERID='"+str(uid)+"' AND SID='"+str(sid)+"'"
276         rows = fetch(field, table, clause)
277         CSV_FILE = open(dirSave+str(uid)+"_"+str(sid)+".csv", "w")
278         for row in rows:
279             CSV_LIST = []
280             CV(row['EID'])
281             CV(row['TTYPE'])
282             CV(row['ACTION'])
283             getValue(row, uid, sid)
284             timestamp(row['TTIME'], row['MS'], row['ACTION'])
285             if row['TTYPE'] not in EX_EXTRA:
286                 CV(row['RELATION'])
287                 getExtra(row, uid, sid)
288                 CSV_FILE.write(WRITE())
289             CSV_FILE.close()
290             TENDFILE = datetime.now()
291             LOG.write("TIME_SPENT: "+str((TENDFILE-TSTARTFILE))+"\n")
292             LOG.write("TIME_END: "+str(TENDFILE)+"\n")
293 TEND = datetime.now()
294 LOG.write("TIME_SPENT_TOTAL: "+str((TEND-TSTART))+"\n")
295 LOG.write("TIME_END: "+str(TEND)+"\n")
296 LOG.close()

```

### 3.7 Export CSV to CSV

The CSV format we get from Syslog-NG is not exactly like it's supposed to be. The timestamps must be transformed to number of milliseconds since the first event, not full date and time. The system-messages from BeLT are left untouched, but the remaining messages must be transformed to number of milliseconds since the first "start" message from BeLT.

To export a file, you call the script, where the first parameter is the file that should be exported, like this:

```
1 /var/log/belt/csv2csv.py ID/ID_1.csv
```

By default the file generated will have the the same filename, except, they have ”\_formatted” before the file extension, which should be ”csv”. The full

Listing 9: csv2csv.py file

```
1 #!/usr/bin/env python
2 # Transforms all the timestamps in the csv file to milliseconds
3 # since start. All BeLT system messages are left untouched.
4
5 import os, sys, math
6 import re, time
7 import string, random, os
8
9 # We also change the format to UTF-8
10 sourceEncoding = "iso-8859-1"
11 targetEncoding = "utf-8"
12
13 # Where the position of the time field is
14 TIMEFIELDREG = 4
15 TIMEFIELDSCR = 8
16 TIMEFIELDSCR2 = 4
17 TIMEFIELDKEY = 5
18 TIMEFIELDDEV = 4
19 TIMEFIELDRES = 5
20
21 def id_generator(size=6, chars=string.ascii_uppercase + string.
    digits):
22     return ''.join(random.choice(chars) for x in range(size))
23
24 # Get UNIX epoch time from a ISODATE string
25 def getMsTime(syslogTime):
26     splitDate = syslogTime.split('.')
27     seconds1 = time.strptime(splitDate[0], "%Y-%m-%dT%H:%M:%S")
28     seconds = int(time.mktime(seconds1))
29     splitTime = splitDate[1].split('+')
30     ms = seconds * 1000
31     ms = ms + int(splitTime[0])
32     return ms
33
34 if(len(sys.argv) < 2):
35     print "Usage: "+sys.argv[0]+" <CSV file>"
36     exit(1)
37
38 fileExt = "csv"
39 outName = ""
40 outFile = ""
41 filename = sys.argv[1]
42
43 # Check if the user supplied an output name
44 # If not we use (filename - extension) + "_formatted" + extension
45 if(len(sys.argv) > 2):
46     outFile = sys.argv[2]
47 else:
```

```

48     outName, fileExt = os.path.splitext(filename)
49     outFile = outName + "_formatted" + fileExt
50
51 # Convert the whole file to UTF-8
52 f = open(filename)
53 tmpFile = id_generator(15)
54 tmpFile += ".utf8.tmp"
55 b = open(tmpFile, "w")
56 b.write(unicode(f.read(), sourceEncoding).encode(targetEncoding))
57 b.close()
58 f.close()
59
60 # Open the file for writing
61 File = open(outFile, 'w')
62
63 # Read all the lines from our temporary file
64 lines = [line.strip() for line in open(tmpFile)]
65
66 os.remove(tmpFile)
67
68 beginning = 0
69 for line in lines:
70     # Default for everything except BeLT messages and Hardware
71     currTime = TIMEFIELDREG
72     fields = line.split(',')
73     if(fields[1] == 'B'): # Event from BeLT
74         if(fields[2] == 'start'): # Calcucate start time
75             beginning = getMsTime(fields[3])
76     else:
77         if(fields[2] == 'SCR_Info'):
78             currTime = TIMEFIELDSCR
79         elif(fields[2] == 'SCR'):
80             currTime = TIMEFIELDSCR2
81         elif(fields[2] == 'KEY'):
82             currTime = TIMEFIELDKEY
83         elif(fields[2] == 'RES'):
84             currTime = TIMEFIELDRES
85         elif(fields[2] == 'DEV'):
86             currTime = TIMEFIELDDEV
87         newTime = getMsTime(fields[currTime]) - beginning
88         fields[currTime] = str(newTime)
89     # Write all fields back to the file
90     File.write(",".join(fields))
91     File.write('\n')

```

### 3.8 Bugzilla

Bugzilla are dependent on Apache, MySQL, Perl and PHP. When visiting their webpage you see their minimum okay version of dependencies to different version releases. We use Bugzilla version 4.2.4 on an Ubuntu 12.04 LTS server. This is the command that install all packages needed to run Bugzilla:

```

1 apt-get install libapache-mod-perl2 tasksel libyaml-perl gcc make
  vim sendmail

```

To install a LAMP-server (Linux-Apache-MySQL-Perl/PHP/Python) on Ubuntu you type

```
1 tasksel install lamp-server
```

and follow the dialog. The ftp root of all Bugzilla releases are located at `ftp://ftp.mozilla.org/pub/mozilla.org/webtools/`. On the server we would do the following to download extract and the untar tarball of Bugzilla.

```
1 cd /var/www/
2 wget ftp.mozilla.org/pub/mozilla.org/webtools/bugzilla-4.2.4.tar.gz
3 tar -xvzf bugzilla-4.2.4.tar.gz
```

Then you rename the folder as you want, we chose to call it 'bugs'. Apache has a separate file to apply additional configuration than the one that is provided, 'httpd.conf' located in '/etc/apache2/'. We added this configuration to our bugs directory to have apache display the bugtracker:

```
1 <Directory /var/www/bugs>
2 AddHandler cgi-script .cgi
3 Options +Indexes +ExecCGI
4 DirectoryIndex index.cgi
5 AllowOverride Limit FileInfo Indexes
6 </Directory>
```

CGI, common gateway interface, is a service on the server that executes the files to generate web content on the site. The cgi is based on perl in our case. The code above should set this option to happen. When you have done this you should also set the option to execute CGI's in from your root web-directory in the default file located at '/etc/apache2/sites-enabled/'. It should look very similar to the code sample, just append '+ExecCGI' after 'Options':

```
1 ...
2 <Directory /var/www/>
3 Options Indexes +ExecCGI FollowSymLinks MultiViews
4 AllowOverride None
5 Order allow,deny
6 allow from all
7 </Directory>
8 ...
```

After you have done this you have all dependencies ready to install Bugzilla, follow the official guide for your chosen version at <http://www.bugzilla.org/docs/>, the documentation for the 4.2-series is found at <http://www.bugzilla.org/docs/4.2/en/html/>.

### 3.8.1 Relevant experiences from our installation on Ubuntu

In our case the virtual server had the following ports changed with NAT:

Protocol	Std. port	Used port
SSH	22	19150
HTTP	80	19151
HTTPS	443	19152
Syslog-NG	1999	19155

Some experiences we had was:

- In the local config-file, 'webservers-group' with default installation of apache should be changed to 'www-data'.
- Because of the virtual server and the routing of ports we had problems with the default Sendmail-configuration. We ended up with using a CPAN-extension.

To fix the problem with sendmail in Bugzilla, we installed a module found at <http://search.cpan.org/~lbroadcard/Email-Send-Gmail-0.33/lib/Email/Send/Gmail.pm> by issuing the following perl commands.

```
1 perl -MCPAN -e shell
2 cpan[1]> install Email::Send::Gmail
3 cpan[2]> quit
```

We don't have a detailed explanation to why this worked, since the bug-tracker was just a method of complementing our ongoing work. To change mail configuration you go to the Bugzilla homepage, click 'Preferences' from the top and then 'Email preferences'. We'd like to thank the creator of this fix <http://rndm-snippets.blogspot.no/2011/05/bugzilla-issue-tracking-send-mail.html> for the sharing it with the community.

### 3.9 Import and export of Bugzilla databases

To export the contents of the information from Bugzilla we started out by wanting just the data, but this would most likely cause massive problems if not absolutely all the data was gathered. Therefore we decided to export the entire MySQL database from the bugtracker database, "bugs".

To do this one needs access to the servers, have complete access to the databases, since we need to export the database and then later on import it again. Then one needs to have a MySQL server on both servers and the tool "mysqldump" which is used to dump the database to a textual "sql" file.

#### Export

Use the root user or another user with the necessary privileges to dump the database to an SQL formatted file by issuing the command:

```
1 mysqldump -p -u root dbname > "/path/to/dumpfile.sql"
```

**-p:** argument means that the user has to provide a password.

**-u:** argument means the following word is the name of the user.

**root:** this is the users username and could be any user with the correct privileges.

**dbname:** is the name of the database you are exporting to an sql file.

**>:** means you redirect the output to file where the following word encapsulated with two " is the absolute or relative path to the file.

**Import:**

After moving the stored file to the new server one has to import the new file into the new database.

```
1 mysql -p -u root dbname < "path/to/dumpfile.sql"
```

**-p:** Means that the user has to provide a password.

**-u:** specifies the user to log in as in the following word.

**root:** this is the users username and could be any user with the correct privileges.

**dbname:** the name of the new database to import into.

**<:** means the content of the following file will be sent as standard input to the mysql process.

**Creating a user:**

After importing the database to the MySQL server one should create a new user to avoid having the root user manage the database and grant the new user only access to the bugtracker database.

To create the user for the bugtracker system use the following command:

```
1 GRANT ALL PRIVILEGES
2   ON dbname.*
3   TO 'username'@'localhost'
4   IDENTIFIED BY 'password';
```

**GRANT ALL PRIVILEGES:** Lets the user have all privileges except the GRANT option for the database.

**ON dbname.\*:** Specifies which database and tables to grant the user access to. The database is identified by 'dbname'. The databases tables are either specified by their name or the asterisk symbol to grant the user privileges to all the tables in the database.

**TO:** means the following "username"@localhost" gives the user only access locally on the server.

**IDENTIFIED BY:** means the user gets assigned the following password within the apostrophes.

## 4 Software Distribution

Here we will describe the scripts that does our software distribution and package management. We have also provided some extra theory about the package management because it is very nice to have a place to start learning about this, knowledge about this is important for being able to deploy software.

### 4.1 Deploy.ps1 - main script

The deploy.ps1 flowchart is shown in figure 2. It starts with **(1)** generating the source code documentation. After this it will **(2)** call the script that builds binaries (build.ps1) based on the platform (-p) input parameters 'x64' and/or the 'x86' (it builds both on default). When the binaries are ready it will **(3)** run the wix.ps1, the script that controls our package management and code signing using WiX 4.2. Wix.ps1 packages the files into MSI (installer) or MSP (patch) files, it also also codesigns the compiled binaries and the following MSI/MSPs to ensure that they are created by us and nobody else. At last **(4)** the script archives the source code with WinRAR.exe and moves the archive to a new branch based on the version number.

If the local(-l) parameter was set we upload the entire branch to the repository so we have a backup of our source code.

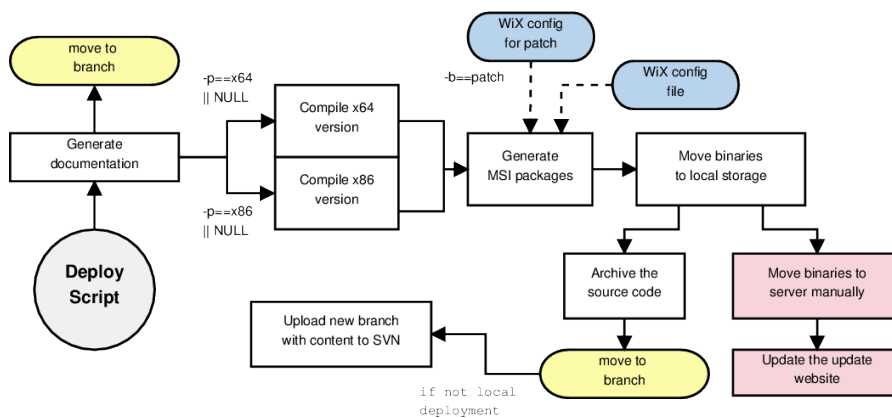


Figure 2: Flowchart for our deployment process.

### 4.2 Windows Installer XML toolset(WiX)

#### WiX configuration

The XML configuration file has a few elements that has to be included:

- <wix> is the root element in the XML file.

- `<product>` defines the product, and also has many attributes that play a role
- `<directory>` you have to include a at least one directory, namely the `installdir`, where you place the main executables of the program. You can also provide folders of different locations to put several files. Common directories are `appdir`, `programdir` and `programfiles`.
- `<component>` it is like a function, you tell it to do something, the many varieties for this are found in the manual. We used the component element to make registry entries, use `mergemodules` and for creating shortcuts.
- `<feature>` here you list the components that should be executed by wix, its like a list saying "do this".

Inside these elements you have to declare at least an ID and a name. Components need a unique ID, and to ensure uniqueness we have generated GUID's. If you are going to create a hierarchical structure of directories and files, WiX can do this when you nest the elements, for example declaring a directory within another. We have commented our configuration files and divided them into codeblocks so it's easier to add, edit or remove specific parts of them.

### wix.ps1

When releasing software you have two options to choose from, you make a software upgrade/update or a patch. Microsoft has put clear definitions to this. Long story short, a patch is the binary difference between two installers (it only applies the difference made) whereas an upgrade removes the old installation first and then installs the new one. The most useful aspect of this approach is in that when your users download these files, a patch will be a lot smaller than the upgrade, depending on the changes that has happened. A patch has one weakness, and that is that you can't introduce new files on your install (you can only change parts of existing program files).

Upgrades need a new Product ID each time. A patch continues on the last upgrade, using the same product ID as that version.

When calling the script from PowerShell, you can send with parameters to choose if you want to build an upgrade or a patch. The current setup works with WiX 3.7, and until now (05.04.2013) we have no known errors or wrong behaviour.

The part of building an upgrade is divided into a two stage compilation.

1. `candle.exe` (part of the WiX Toolset) to make WiX object files.
2. `light.exe` (part of the WiX Toolset) to interpret these object files and convert them into the Windows Installer format.

First it parses the XML configuration and binaries to WiX-objects, and then builds a windows installer package based on these objects. It is really intuitive when you think about it, first make the candle, and then let it be light!



After we finish this part, we move the files into folders based on the version number, and then perform an administrative install locally. The administrative installation is a special way of extracting all the contents in the Windows Installer package that normally wouldn't exist after an actual installation. These files are crucial and important information when you are making the patches and want to compare two different ones.

You can see that we always make an upgrade in our script. This is because when we are making patches we need two different versions samples to compare. Wix.ps1 works in this sequence:

1. Sign files
2. Make upgrade for the newest version
3. Move the installer and all files to a folder
4. Perform Administrative installation
5. Sign newly made files
6. (Make a patch)

Creating a patch is a more complicated operation than creating an upgrade. We can divide the process of creating a patch into 3 parts:

1. First we use Torch.exe, it creates WiX object files (wixmst) in the difference between two installs
2. Candle/light, creates a patch baseline from your configuration files (patch.wxs) -> (wixmsp)
3. pyro.exe, merges/joins the wixmst and wixmsp into a msp-file (Microsoft patch file)

### 4.3 Codesigning

We encourage the reader to briefly know what the different file formats contain. pfx and pem carry the same content, but they have a different structure.

- pfx - Microsoft proprietary format, contains key and certificate
- pem - certificate and the key
- crt - only the certificate
- key - the key

When you want to use the codesign.exe you need to provide a pfx file. Very often when you apply for certificates, you get a pem file, or a crt and key file. It is possible to convert between these formats with the OpenSSL utility. We used the following command in PowerShell:

Listing 10: "Command to sign a file, using signtool.exe"

```

1 # Convert the pem to pfx
2 > openssl.exe pkcs12 -export '
3 -out certificate.pfx
4 -inkey .\code_sign.key '
5 -in .\cert-3108-Høgskolen_i_Gjøvik.pem '
6 -certfile .\chain-3108-Høgskolen_i_Gjøvik-3-
   AddTrust_External_CA_Root.pem

```

Listing 11: "Command to sign a file, using signtool.exe"

```

1 # Sign, f->certificate, p->password,
2 #     t->timestamp .. [file to sign]
3 > signtool.exe sign /f .\certificate.pfx '
4 -p [PASSWORD] '
5 /t http://timestamp.verisign.com/scripts/timestamp.dll '
6 /d "BeLT" .\belt_installer.msi

```

### Script routine

Microsoft needs pfx, that has a different structure. We use openssl to convert it into this, and then we use Microsoft's codesigning tool codesign.exe. We need to provide a key with a timestamp, and then its okay.

## 4.4 WiX build-script

Listing 12: "Script to generate the MSI package"

```

1 #####
2 # WiX build script
3 #####
4 # This script controls all package management action
5 # performed by WiX
6 #
7 # Parameters:
8 # -o for the old version
9 # -n for the new version
10 # -p for the platform (x64 or x86), defaults to x64
11 # -b for build options (supports only "patch")
12 #####
13
14 Param(
15     [parameter(Mandatory=$false)]
16     [alias("o")]
17     $oldVersion,
18
19     [parameter(Mandatory=$true)]
20     [alias("n")]
21     $newVersion,
22
23     [parameter(Mandatory=$false)]
24     [alias("b")]
25     $build ,
26

```

```

27 [parameter(Mandatory=$false)]
28 [alias("p")]
29 $platform="x64",
30
31 [parameter(Mandatory=$true)]
32 [alias("x")]
33 $password,
34
35 [parameter(Mandatory=$true)]
36 [alias("g")]
37 $guid
38 )
39
40
41 # PATH VARIABLES
42 $scriptpath = $MyInvocation.MyCommand.Path
43 $dir = Split-Path $scriptpath
44
45 signtool.exe sign /f .\certificate.pfx -p $password /t http://
    timestamp.verisign.com/scripts/timestamp.dll /d "BeLT" .\belt_main
    .exe .\belt_update.exe
46
47 if(!(Test-Path ("{$dir\$newVersion\$platform}"))) {
48   & candle.exe -dPlatform="$platform" -dVersion="$newVersion" -
    dGuid="$guid" $dir\belt_installer.wxs
49   & light.exe -ext WixUIExtension $dir\belt_installer.wixobj
50 }
51 else { write-host "There already made an installer for version
    $newVersion on the platform $platform" }
52
53 signtool.exe sign /f .\certificate.pfx -p $password /t http://
    timestamp.verisign.com/scripts/timestamp.dll /d "BeLT" .\
    belt_installer.msi
54
55 if (!(Test-Path ("{$dir\$newVersion\$platform}"))) {
56   New-Item -Type Directory -Path $newVersion\$platform
57   # New-Item -Type Directory -Path ..\..\branches\$newVersion\
    $platform
58   # cp belt_installer.msi ..\..\branches\.$newVersion\$platform
59   mv belt_installer.wixpdb .\$newVersion\$platform
60   mv belt_installer.wixobj .\$newVersion\$platform
61   mv belt_installer.msi .\$newVersion\$platform
62   mv belt_main.exe .\$newVersion\$platform
63   mv belt_update.exe .\$newVersion\$platform
64   #mv dllhook.dll .\$newVersion\$platform
65
66   # Administrative installation, used for creating patches
67   $endDir="$dir\$newVersion\$platform\admin"
68   $installprop = "TARGETDIR=" + "''" + $endDir + "''"
69   Write-Host $installprop
70   invoke-expression "msiexec.exe /a $newVersion\$platform\
    belt_installer.msi $installprop"
71 }
72 echo "I am taking a nap for 15 sec, please click away the
    installation dialoges"
73 Start-Sleep -s 15
74 if($build -eq "patch") {

```

```

75 Write-host "I will build you the best patch there is"
76 torch.exe -p -ax debug -xo $oldVersion/$platform/admin/
    belt_installer.msi $newVersion/$platform/admin/belt_installer.
    msi -out patch.wixmst
77 candle.exe patch.wxs -dVersion="$newVersion"
78 light.exe patch.wixobj -out patch.wixmsp
79 pyro.exe patch.wixmsp -out patch.msp -t RTM patch.wixmst
80 signtool.exe sign /f .\certificate.pfx -p $password /t http://
    timestamp.verisign.com/scripts/timestamp.dll /d "BeLT" .\patch.
    msp
81 New-Item -Type Directory -Path $newVersion\$platform\patch
82 mv patch.msp .\$newVersion\$platform\patch
83 }
84
85
86 # Moving all files to a timestamped directory
87 # $timestamp = Get-Date -f yy.MM.dd_hh.mm.ss #hh.mm.ss_dd.MM.yy
88 #New-Item -Type Directory -Path $timestamp

```

## 4.5 WiX configuration file

Listing 13: "WiX XML configuration file"

```

1 <?xml version="1.0"?>
2
3 <!--
4 NEVER EVER CHANGE PRODUCT UPGRADECODE
5 -CHANGE THE PRODUCT ID WHEN DOING MAJOR UPGRADES
6 -STAR GIVES RANDOM NEW PRODUCT ID
7 -->
8 <?define ProductID = "$(var.Guid)" ?>
9
10 <Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
11
12 <!-- NEVER CHANGE THE UPGRADECODE! -->
13 <?define ProductUpgradeCode = "a54ea86c-863b-4aa5-8eb2-
    cca7aa240973"?>
14 <!-- PLATFORM DEPENDENT SETTINGS -->
15 <?if $(var.Platform)= x64 ?>
16 <?define win64Flag = "yes" ?>
17 <?else ?>
18 <?define win64Flag = "no" ?>
19 <?endif ?>
20
21 <!-- PATH TO MERGEFILES AND DDL'S -->
22 <?define MergeDir = "C:\Program Files (x86)\Common Files\Merge
    Modules" ?>
23 <?define SSL86Dir = "C:\OpenSSL-Win32" ?>
24 <?define SSL64Dir = "C:\OpenSSL-Win64" ?>
25
26 <Product
27 Id="$(var.ProductID)"
28 UpgradeCode="$(var.ProductUpgradeCode)"
29 Name="Behavior Logging Tool" Version="$(var.Version)"
    Manufacturer="NISLab" Language="1033">
30 <Package InstallerVersion="500" Compressed="yes" Platform="$(var.
    Platform)" Comments="Windows Installer Package" />

```

```

31 <Media Id="1" Cabinet="product.cab" EmbedCab="yes"/>
32 <Icon Id="ProductIcon" SourceFile="key_512.ico"/>
33 <Upgrade Id="$(var.ProductUpgradeCode)">
34   <UpgradeVersion Minimum="$(var.Version)" OnlyDetect="yes"
      Property="NEWERVERSIONDETECTED"/>
35   <UpgradeVersion Minimum="0.0.0" Maximum="$(var.Version)"
      IncludeMinimum="yes" IncludeMaximum="no"
      Property="OLDERVERSIONBEINGUPGRADED"/>
36 </Upgrade>
37 <Condition Message="A newer version of this software is already
38   installed.">NOT NEWERVERSIONDETECTED</Condition>
39
40 <!-- Creating product, dedicates files to the main folder and
41   merge objects
42   Uses if-sentence to determine platform, platform is defined in
43   config.wxi-->
44 <Directory Id="TARGETDIR" Name="SourceDir">
45   <?if $(var.Platform) = x86 ?>
46     <Directory Id="ProgramFilesFolder">
47       <Directory Id="INSTALLDIR" Name="belt">
48         <Component Id="ApplicationFiles" Guid="aca4037f-9ee6-4b86
49           -98ec-335ff383778a">
50           <File Id="ApplicationFile1" Source="belt_main.exe"
51             KeyPath='yes' />
52           <File Id="ApplicationFile2" Source="dllhook.dll"/>
53           <File Id="ApplicationFile3" Source="$(var.SSL86Dir)\
54             ssleay32.dll"/>
55           <File Id="ApplicationFile5" Source="$(var.SSL86Dir)\
56             libeay32.dll"/>
57           <File Id="ApplicationFile6" Source="belt_update.exe"/>
58         </Component>
59         <Merge Id="Microsoft_VC110_CRT_x86" SourceFile="$(var.
60           MergeDir)\Microsoft_VC110_CRT_x86.msm" Language="0"
61           DiskId="1"/>
62         <Merge Id="Microsoft_VC110_MFC_x86" SourceFile="$(var.
63           MergeDir)\Microsoft_VC110_MFC_x86.msm" Language="0"
64           DiskId="1"/>
65       </Directory>
66     </Directory>
67   <?else ?>
68     <Directory Id="ProgramFiles64Folder">
69       <Directory Id="INSTALLDIR" Name="belt">
70         <Component Id="ApplicationFiles" Guid="3c40bf41-22f5-4c4d
71           -9a40-204c8992aa69">
72           <File Id="_64ApplicationFile1" Source="belt_main.exe"
73             KeyPath='yes' />
74           <File Id="_64ApplicationFile2" Source="dllhook.dll
75             "/> -->
76           <File Id="_64ApplicationFile3" Source="$(var.SSL64Dir)\
77             ssleay32.dll"/>
78           <File Id="_64ApplicationFile5" Source="belt_update.exe
79             "/>
80           <File Id="_64ApplicationFile6" Source="$(var.SSL64Dir)\
81             libeay32.dll"/>
82           <File Id="_64ApplicationFile7" Source="README.txt
83             "/> -->

```

```

67     </Component>
68     <Merge Id="Microsoft_VC110_CRT_x64" SourceFile="$(var.
        MergeDir)\Microsoft_VC110_CRT_x64.msm" Language="0"
        DiskId="1"/>
69     <Merge Id="Microsoft_VC110_MFC_x64" SourceFile="$(var.
        MergeDir)\Microsoft_VC110_MFC_x64.msm" Language="0"
        DiskId="1"/>
70     </Directory>
71 </Directory>
72 <?endif ?>
73
74 <!-- Puts settings.ini inside the programdata, this folder has a
75     special ACL,
76     so the user will be unable to alter the settings without
77     elevated privileges -->
78 <Directory Id="CommonAppDataFolder">
79     <Directory Id="BeltAppData" Name="NISLab">
80     <Directory Id="BeltAppData2" Name="belt">
81     <Component Id="BeltConfig" Guid="751979C2-F264-4FA4
        -8144-72ADF7754B73">
82     <File Id="ConfigFile" Source="..\belt_main\settings.ini
        "/>
83     <File Id="ConfigFile1" Source="..\..\..\
        Additional_Material\CA-chain.pem"/>
84     </Component>
85     </Directory>
86 </Directory>
87 </Directory>
88 <!-- Creating the shortcut -->
89 <Directory Id="ProgramMenuFolder">
90 <!-- <Directory Id="ProgramMenuSubfolder" Name="BeLT" --> -->
91 <Component Id="ApplicationShortcuts" Guid="35efd5ff-cd8b-4ddc
        -b827-1525ed8f3d51" Win64="$(var.win64Flag)">
92 <?if $(var.Platform) = x86 ?>
93 <Shortcut Id="ApplicationShortcut1" Name="BeLT"
        Description="Belt"
94     Target="[INSTALLDIR]belt_main.exe" WorkingDirectory="
        INSTALLDIR"/>
95 <?else ?>
96 <Shortcut Id="ApplicationShortcut2" Name="BeLT"
        Description="Belt"
97     Target="[INSTALLDIR]belt_main.exe" WorkingDirectory="
        INSTALLDIR"/>
98 <?endif ?>
99 <RegistryValue Root="HKLM" Key="SOFTWARE\NISLab\belt"
        Name="installed" Type="integer" Value="1" KeyPath="yes"
        />
100 <RemoveFolder Id="ProgramMenuSubfolder" On="uninstall"/>
101 </Component>
102 <!-- </Directory> -->
103 </Directory>
104 </Directory>
105
106 <DirectoryRef Id="TARGETDIR">
107 <Component Id="RegistryEntries" Guid="2a381431-7299-491e-b3e7-
        bf89a1534163" Win64="$(var.win64Flag)">

```

```

108     <RegistryKey Root="HKLM" Key="SOFTWARE\NISLab\belt">
109         <RegistryValue Type="string" Name="Version" Value="$(var.
            Version)"/>
110     </RegistryKey>
111 </Component>
112 </DirectoryRef>
113
114     <InstallExecuteSequence>
115         <RemoveExistingProducts After="InstallValidate"/>
116     </InstallExecuteSequence>
117
118 <!-- Defines the components to be executed -->
119 <Feature Id="DefaultFeature" Level="1">
120     <ComponentRef Id="ApplicationFiles"/>
121     <ComponentRef Id="ApplicationShortcuts"/>
122     <ComponentRef Id="BeltConfig" />
123     <ComponentRef Id="RegistryEntries" />
124     <?if $(var.Platform) = x86 ?>
125         <MergeRef Id="Microsoft_VC110_CRT_x86"/>
126         <MergeRef Id="Microsoft_VC110_MFC_x86"/>
127     <?else ?>
128         <MergeRef Id="Microsoft_VC110_CRT_x64"/>
129         <MergeRef Id="Microsoft_VC110_MFC_x64"/>
130     <?endif ?>
131 </Feature>
132
133 <!-- GUI Parameters -->
134 <Property Id="WIXUI_INSTALLDIR" Value="INSTALLDIR" />
135 <UIRef Id="WixUI_InstallDir" />
136 <WixVariable Id="WixUILicenseRtf" Value="license.rtf" />
137
138 </Product>
139 </Wix>

```

## 4.6 WiX patch configuration file

Listing 14: "WiX patch configuration file"

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
3     <Patch Classification="Update" Description="Small Update Patch"
4         DisplayName="Patch to version "
5         MoreInfoURL="http://www.nislab.no" Manufacturer="NISLab"
6         AllowRemoval="yes">
7         <Media Id="5000" Cabinet="product.cab">
8             <PatchBaseline Id="RTM">
9                 <Validate ProductVersionOperator="Equal" ProductVersion="
10                     Update" ProductId="no"></Validate>
11             </PatchBaseline>
12         </Media>
13     </Patch>
14     <PatchFamilyRef Id="BeltPatch"></PatchFamilyRef>
15 </Patch>
16 <Fragment>
17     <PatchFamily Id="BeltPatch" Supersede="yes" Version="$(var.
18         Version)"/>
19 </Fragment>
20 </Wix>

```





# BELT: Behavior Logging Tool User manual

**Patrick Bours** `patrick.bours@hig.no`  
**Soumik Mondal** `soumik.mondal@hig.no`



(Behaviour Logging Tool)



(Norwegian Information Security laboratory)

May 15, 2013

## User manual

If you are running Windows 7 or 8, press the windows key on you keyboard and type "belt" – you should now see the shortcut for BeLT, click to run. When BeLT is run for the first time it will go hide in the system tray (lower right corner of you screen). As you see in this picture, the indicator icon for BeLT is the B with a green circle around it.



Figure 1: BeLT is logging and running

Right after installation BeLT might not be visible directly in the tray. Like this:



Figure 2: BeLT is gone

BeLT is actually not gone, it's just in Windows tray hiding mode. You can gain BeLT in the tray by clicking 'Customize' after clicking on the arrow aiming high:

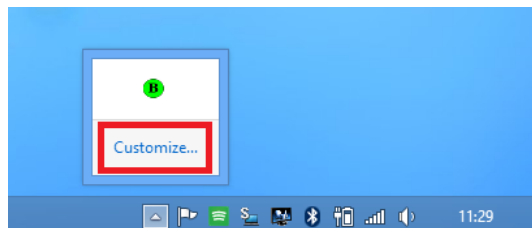


Figure 3: When you are customizing the taskbar, you can choose what you want visible or not

### Not logging password fields

Belt has the ability to filter out passwords. It is not 100% reliable but it will filter out passwords from Firefox and Internet Explorer, and also programs in Windows that sensor passwords. When BeLT chooses to do this, it will indicate this by turning yellow:



Figure 4: BeLT in password filtering mode

You can right click this icon to choose actions for controlling the behaviour of BeLT. If you try to right click, you can restore, stop, pause and exit the program. The difference between pause and stop is that pause keeps you session but stop to log – the stop button quits both your session and logging. Here is an example where a user has right clicked the BeLT icon, and is ready to restore BeLT.

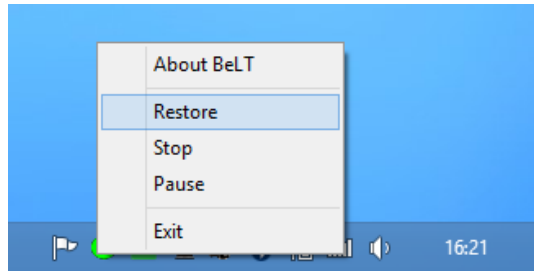


Figure 5: Restore GUI from tray

The color of the BeLT icon changes according to the logging status of BeLT, it is yellow if has detected a password and red if it's stopped. Here is an example of a stopped BeLT:



Figure 6: BeLT is not logging, but running

And this is the BeLT interface after being brought up from the tray.

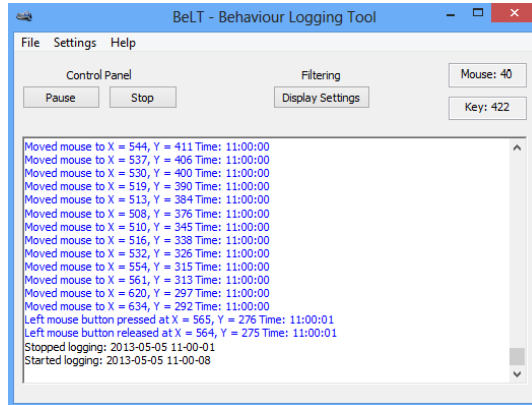


Figure 7: BeLT interface after restoring

As you can see, there is a big field in the middle – this is the place that is used to visualize data. If you click the "Display Settings" button, a lot of checkboxes will appear. Like this:

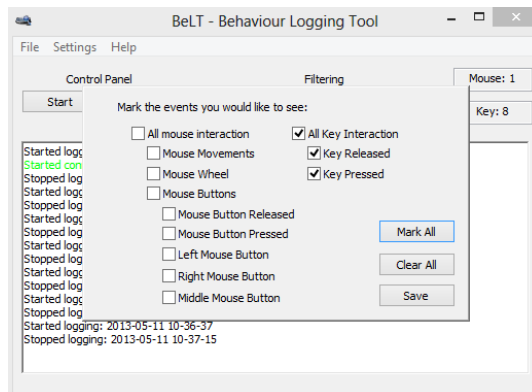


Figure 8: You can check and uncheck boxes to visualize logging behavior

If you want to change advanced settings of BeLT, click the settings in the top row. As you see below, a user is about to select the settings settings:

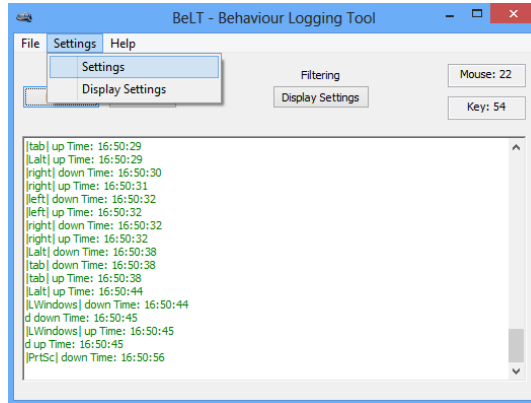


Figure 9: In this place you can adjust advanced settings

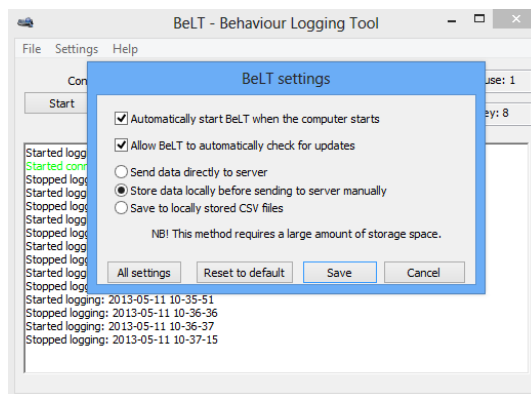


Figure 10: Advanced settings

If you want to change the core behaviour of BeLT, you just found the place to do it. Here you can choose to store the logged data locally on your computer, if you choose to do this, there is a way to send the information to the server afterwards.

If you choose to send the information as local files, you can look through your data before sending it. This is a tedious way, but it gives you more control of your data if you like that. This is a user about to use this ability:

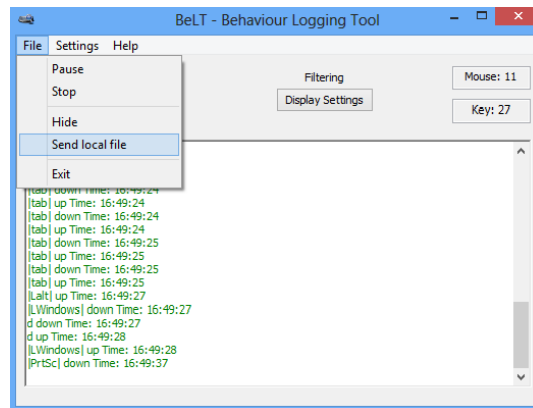


Figure 11: This setting enables you to send a local file

This thing does not give you insight of the contents of you file(s), but you can locate every file inside a folder in your Documents folder, named with your unique ID (a long string with random numbers and letters).

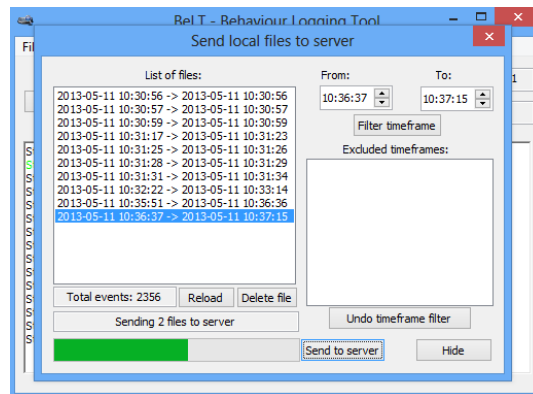


Figure 12: Choose session file or a timeframe, click send

# Troubleshooting

## Known problems

### Application exits on start

One of F-Secure 2011 earlier versions of the DeepGuard system may cause BeLT not to start. This is not a problem caused by BeLT, but by F-Secures protection system "DeepGuard". DeepGuard will run a more restrictive guard of BeLT since it is unknown to DeepGuards hash library. This guard may cause an exception to be thrown when BeLT is connecting to the Internet. The exception will then cause BeLT to exit prematurely. Since this is a standard Windows function call to establish a socket we are unable to disable this problem, but F-Secure recommends to update to a newer version of F-Secure, with a newer version of DeepGuard.

BeLT may at the time of exit have already created a shortcut inside the startup-folder for Windows which will cause it to start automatically each time Windows starts. Which will in return cause BeLT to exit each time the system is started. To fix this problem, uninstall BeLT before exiting Windows.

If this problem occurs, try downloading a newer version/upgrade your F-Secure appliance. The problem was discovered on an 2011 version of F-Secure which incorporated one of the earliest versions of DeepGuard. Visit F-Secures website, [www.f-secure.com/](http://www.f-secure.com/), and forums to get help on-line from their technical support and community.

### BeLT being detected as accessibility application

Some applications, like Adobe Reader will detect BeLT as an accessibility application. Adobe Reader will then want you to set up Adobe Reader so that it is better supported for Accessibility applications. This setup is fast and easy to go through.

## Error reporting

Any problems is to be reported to Prof.Dr. Patrick Bours at [patrick.bours@hig.no](mailto:patrick.bours@hig.no) or Mr. Soumik Mondal at [soumik.mondal@hig.no](mailto:soumik.mondal@hig.no), project managers for BeLT, at NISlab<sup>1</sup>. Please e-mail a detailed description of what you where doing while the error/problem occurred along with a description of the error.

---

<sup>1</sup>NISlab: Norwegian Information Security laboratory. [www.nislab.no](http://www.nislab.no)





## C Windows application certification

<b>Checklist for windows app certification requirements</b>	
<b>Description</b>	<b>Check</b>
<b>1 – Compatibility</b>	
- No dependency on compatibility modes	✓
- No dependency on VB6 Runtime	✓
- Must not load arbitrary DLLs to intercept Win32 API calls using ap- pInit_dlls	✓
<b>2 – Security</b>	
<a href="http://msdn.microsoft.com/en-us/library/windows/desktop/aa374872%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/windows/desktop/aa374872%28v=vs.85%29.aspx</a> - 2.1 ACL to secure <b>Executable files</b>	✓
- 2.2 ACL to secure <b>Directories</b>	✓
- 2.3 ACL to secure <b>Registry keys</b>	✓
- 2.4 ACL to secure <b>Directories that contain objects</b>	✓
- 2.5 Your app must reduce non-administrator access to services that are vulnerable to tampering	✓
- 2.6 Your app must prevent services with fast restarts from restarting more than twice every 24 hours	✓
<b>3 – More Security</b>	
- 3.1 Your app must not use AllowPartiallyTrustedCallersAttribute (APTCA) to ensure secure access to strong-named assemblies	✓
- 3.2-4 Compilation flags: SafeSEH, NXCOMPAT and NXCOMPAT	✓
- 3.5 Your app must not Read/Write Shared PE Sections	✓
<b>4 – Restart and behaviour</b>	
- 4.1 Your app must handle critical shutdowns appropriately	✓
- 4.2 A GUI app must return TRUE immediately in preparation for a restart	✓
- 4.3 Your app must return 0 within 30 seconds and shut down	✓
- 4.4 Console apps that receive the CTRL_C_EVENT notification should shut down immediately	✓
- 4.5 Drivers must not veto a system shutdown event	✓
<b>5 – Apps must support a clean, reversible installation</b>	
- 5.1 Your app must properly implement a clean, reversible installation	✓
- 5.2 Your app must never force the user to restart the computer immediately	✓
<i>Continues on the next page</i>	

Description	Check
- 5.3 Your app must never be dependent on 8.3 short file names(SFN)	✓
- 5.4 Your app must never block silent install/uninstall	✓
- 5.5 Your app installer must create the correct registry entries to allow successful detection and uninstall	✓ <sup>1</sup>
<b>6 – Apps must digitally sign files and drivers</b>	
- 6.1 All executable files (.exe, .dll, .ocx, .sys, .cpl, .drv, .scr) must be signed with an Authenticode certificate	✓ <sup>2</sup>
- 6.2 All kernel mode drivers installed by the app must have a Microsoft signature obtained through the Windows Hardware Certification program. All File System filter drivers must be signed by Microsoft.	✓
- 6.3 Exceptions and Waivers	✓
<b>7 – Apps don't block installation or app launch based on an operating system version check</b>	
- 7.1 Your app must not perform version checks for equality	✓
- 7.2 Exceptions and Waivers will be considered for apps meeting the criteria below: <ul style="list-style-type: none"> <li>• Apps that are delivered as one package that also run on Windows XP, Windows Vista, and Windows 7, and need to check the operating system version to determine which components to install on a given operating system.</li> <li>• Apps that check only the minimum version of the operating system (during install only, not at runtime) by using only the approved API calls, and that properly list the minimum version requirement in the app manifest.</li> <li>• Security apps (antivirus, firewall, etc.), system utilities (for example, defrag, backups, and diagnostics tools) that check the operating system version by using only the approved API calls.</li> </ul>	✓
<b>8 – Apps don't load services or drivers in safe mode</b>	
- 8.1 Exceptions and Waivers	✓
<b>9 – Apps must follow User Account Control guidelines</b>	
- 9.1 Your app must have a manifest that defines execution levels and tells the operating system what privileges the app requires in order to run	✓
- 9.2 Your app's main process must be run as a standard user (asInvoker).	✓
- 9.3 Exceptions and Waivers	✓
<b>10 – Apps must install to the correct folders by default</b>	
- 10.1 Your app must be installed in the Program Files folder by default	✓ <sup>3</sup>
- 10.2 Your app must avoid starting automatically on startup	✓
<i>Continues on the next page</i>	

<sup>1</sup>Warning: An optional value 'InstallLocation' is missing or invalid for program Behavior Logging Tool.

<sup>2</sup>Warning: Files from OpenSSL was not digitally signed

<sup>3</sup>Warning: Program Behavior Logging Tool fails due to missing install location.

Description	Check
- 10.3 Your app data, which must be shared among users on the computer, should be stored within ProgramData	✓
- 10.4 Your app's data that is exclusive to a specific user and that is not to be shared with other users of the computer, must be stored in Users\user-name\AppData	✓
- 10.5 Your app must never write directly to the "Windows" directory and or subdirectories	✓
- 10.6 Your app must write user data at first run and not during the installation in "per-machine" installations	✓
- 10.7 Exceptions and Waivers	✓
<b>11 – Apps must support multi-user sessions</b>	
- 11.1 Your app must ensure that when running in multiple sessions either locally or remotely, the normal functionality of the app is not adversely affected	✓
- 11.2 Your app's settings and data files must not persist across users	✓
- 11.3 A user's privacy and preferences must be isolated to the user's session	✓
- 11.4 Your app's instances must be isolated from each other	✓
- 11.5 Apps that are installed for multiple users must store data in the correct folder(s) and registry locations	✓
- 11.6 User apps must be able to run in multiple user sessions (Fast User Switching) for both local and remote access	✓ <sup>4</sup>
- 11.7 Your app must check other terminal service (TS) sessions for existing instances of the app	✓
<b>12 – Apps must support x64 versions of Windows</b>	
- 12.1 Your app must natively support 64-bit or, at a minimum, 32-bit Windows-based apps must run seamlessly on 64-bit systems to maintain compatibility with 64-bit versions of Windows	✓
- 12.2 Your app and its installers must not contain any 16-bit code or rely on any 16-bit component	✓
- 12.3 Your app's setup must detect and install the proper drivers and components for the 64-bit architecture	✓
- 12.4 Any shell plug-ins must run on 64-bit versions of Windows	✓
- 12.5 App running under the WoW64 emulator should not attempt to subvert or bypass Wow64 virtualization mechanisms	✓

Table 1: Summary of Microsoft's app certification reqs.

<sup>4</sup>Warning: An error occurred while performing the testing process.



## D Scripts

### D.1 Python script to calculate time statistics

Code D.1: Python program to calculate time statistics on input file

```
1 #!/usr/bin/env python
2 # print_stat.py
3 # Python script to print difference in time between timestamps
4
5 import math, os, sys
6
7 if(len(sys.argv) < 1):
8     print "Usage: "+sys.argv[0] + " <file>"
9     exit(0)
10
11 # Read all lines
12 infile = open(sys.argv[1], 'rb')
13 lines = infile.readlines()
14 infile.close()
15
16 # Get all lines as integers
17 lines = [int(x) for x in lines ]
18
19 avg = 0
20 count_diff = 0
21 count_all = 0
22 all_diff = []
23 last = 0
24
25 for i in lines:
26     if(count_all == 0):
27         last = i
28         count_diff += 1
29     else:
30         if(i > last):
31             count_diff += 1
32             all_diff.append(i-last)
33             avg += (i-last)
34             last = i
35     count_all += 1
36
37 print "Read " + str(count_all) + " timestamps"
38 print str(count_diff) + " unique timestamps"
39 print "Average difference is " + str(float(sum(all_diff)) / len(all_diff) )
40 print "Smallest difference is " + str(min(all_diff))
41 print "Biggest difference is " + str(max(all_diff))
```

### D.2 Python script to measure mouse compression on file

Code D.2: Python script to test mouse compression on files

```
1 #!/usr/bin/env python
2 # Python script to test different values for mouse
3 # compression.
4
5 # Takes a CSV file, with the format: x,y
6
7 # You can also supply the values which determines
8 # the compression rate
9
```

```

10 import csv, math, os, sys
11
12 DIFF_MAX = 5          # Decrease to provide more accuracy
13 LINE_INCREASE = 10   # Decrease to provide more accuracy
14 DIFF_MIN = 1         # Currently not used
15
16 degrees = 0.0
17
18 # Find the difference between two numbers
19 def difference(a, b):
20     c = 0.0
21     if(b < a):
22         c = a
23         a = b
24         b = c
25     if(a+DIFF_MAX < b):
26         return True
27     return False
28
29 def printThis(thisX, thisY, lastX, lastY, lastWX, lastWY, count):
30     global degrees
31     if(count == 0):
32         return True
33
34     x = math.pow(thisX - lastWX, 2)
35     y = math.pow(thisY - lastWY, 2)
36
37     if( (x + y) > LINE_INCREASE):
38         xx = math.fabs(thisX - lastX)
39         yy = math.fabs(thisY - lastY)
40
41         if( (xx == 0 and yy == 0) or (xx == yy)):
42             return False
43         new_degrees = math.atan2(yy, xx)*180/math.pi
44         if(difference(new_degrees, degrees) == True):
45             degrees = new_degrees
46         return True
47     return False
48
49
50 if(len(sys.argv) < 2):
51     print "Usage: "+sys.argv[0] + " <csv file> [<line min> <min degrees>]"
52     exit(0)
53
54 if(len(sys.argv) > 2):
55     LINE_INCREASE = int(sys.argv[2])
56 if(len(sys.argv) > 3):
57     DIFF_MAX = float(sys.argv[3])
58
59 filename, fileext = os.path.splitext(sys.argv[1])
60 outfilename = filename + 'C' + fileext
61
62 infile = open(sys.argv[1], 'rb')
63 outfile = open(outfilename, 'wb')
64 reader = csv.reader(infile)
65 writer = csv.writer(outfile, delimiter=',', quotechar='"',
66 quoting=csv.QUOTE_NONE)
67
68 # Read in all the data and turn them into integers
69 movList = []
70 for row in reader:
71     row = [ int(x) for x in row ]
72     movList.append(row)
73
74 infile.close()
75
76 count = 0

```

```

77 written = 0
78 last_writ = []
79 for xy in movList:
80     if(count > 0):
81         if(printThis(xy[0], xy[1], movList[count-1][0], movList[count-1][1],
82             last_writ[0], last_writ[1], count) == True):
83             written += 1
84             writer.writerow(xy)
85             last_writ = xy
86     else:
87         written += 1
88         writer.writerow(xy)
89         last_writ = xy
90     count += 1
91 outfile.close()
92 print float(written)/(count-1)

```

### D.3 Python script to paint mouse movements from file

Code D.3: Python script to paint mouse movements on a graph

```

1  #!/usr/bin/env python
2  # mouse_paint2.py
3
4  # A small script to paint an image from a series of
5  # coordinates. Used to paint an image of mouse
6  # movements, can be used to test whether the compression
7  # tactic works or not. CSV file need to have x,y in
8  # this ordar as the first two elements
9
10 import csv
11 from numpy import *
12 import pylab as p
13 import os, sys
14
15 if(len(sys.argv) < 3):
16     print "Usage: " + sys.argv[0] + " <original CSV file> <compressed CSV
17     file>\"
18     + " [X-max Y-max]"
19     exit(0)
20
21 # Get filename without extension
22 # Image get same name as first file
23 filename, fileext = os.path.splitext(sys.argv[1])
24 img_name = filename + ".png"
25
26 maxX = 0
27 maxY = 0
28 if(len(sys.argv) >= 5):
29     maxX = int(sys.argv[3])
30     maxY = int(sys.argv[4])
31
32 picTitle = ""
33
34 if(len(sys.argv) >= 6):
35     picTitle = sys.argv[5]
36
37 # Read the csv file
38 infile = open(sys.argv[1], 'rb')
39 reader = csv.reader(infile); # Read csv file
40 movList = [] # Empty list
41 for row in reader: # Create array of all lines
42     movList.append(row)
43 infile.close()
44
45 # Read the seconds csv file
46 infile1 = open(sys.argv[2], 'rb')
47 reader1 = csv.reader(infile1)

```

```

47 movList1 = []
48 for row in reader1:
49     movList1.append(row)
50 infile1.close()
51
52 # Current array is number of lines lists, we need 2 lists, one for x and
    one for
53 # y before we can create an image out of it.
54 tmp = []
55 tmp2 = []
56 for i in movList:
57     tmp.append(int(i[0]))
58     tmp2.append(int(i[1]))
59
60 fig = p.figure()
61 if(picTitle != ""):
62     fig.suptitle(picTitle)
63
64 # Paint original path
65 p.plot(tmp, tmp2, color='black', lw=2)
66
67 tmp = []
68 tmp2 = []
69 for i in movList1:
70     tmp.append(int(i[0]))
71     tmp2.append(int(i[1]))
72
73 # Paint compressed path
74 p.plot(tmp, tmp2, color='red', lw=2)
75
76 if(maxX != 0 and maxY != 0):
77     p.xlim([0,maxX])
78     p.ylim([0,maxY])
79
80 # Just a hack to get it to work, not very elegant
81 p.yticks([0, 100, 200, 300, 400, 500, 600, 700, maxY])
82 p.xticks([0, 200, 400, 600, 800, 1000, 1200, maxX])
83
84
85 p.gca().invert_yaxis() # Screens have 0 in upper left corner
86 fig.savefig(img_name)

```

## D.4 SQL procedure for inserting data into database

Code D.4: SQL procedure for inserting data

```

1 DROP PROCEDURE if exists log;
2
3 DELIMITER //
4
5 CREATE PROCEDURE log(IN tSIDEKO VARCHAR(15), IN tUID CHAR(35), IN tTIME
    CHAR(30), IN tPNM TEXT, IN tPID CHAR(15), IN tSDT TEXT, IN tMSG TEXT)
6 BEGIN
7     DECLARE dSID INT(11);
8     DECLARE dECO INT(11);
9     DECLARE dEVID CHAR(11);
10    DECLARE c1 CHAR(10);
11    DECLARE c2 CHAR(10);
12    DECLARE c3 CHAR(10);
13    DECLARE c4 CHAR(10);
14    DECLARE t1 TEXT;
15    DECLARE t2 TEXT;
16    DECLARE t3 TEXT;
17
18    set c1 = SUBSTRING_INDEX(SUBSTRING_INDEX(tTIME, '.', -1), 'Z', 1);
19    SET tTIME = CONCAT_WS('.', SUBSTRING_INDEX(tTIME, 'T', 1),
    SUBSTRING_INDEX(SUBSTRING_INDEX(tTIME, 'T', -1), '.', 1));
20    SET dSID = SUBSTRING_INDEX(tSIDEKO, '_', 1);

```



```

21 SET dECO = SUBSTRING_INDEX(tSIDEco, '_', -1);
22 IF tMSG = "" THEN
23     SET tMSG=NULL;
24 END IF;
25
26 IF tSDT LIKE "%belt%" THEN
27     SET tMSG = SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'event=', -1), '',
28     1);
29     IF tMSG LIKE "start" THEN
30         INSERT INTO session (SID, UID, START, MS) VALUES(dSID, tUID, tTIME,
31         c1);
32     ELSE IF tMSG LIKE "stop" THEN
33         UPDATE session SET END=tTIME WHERE UID=tUID AND SID=dSID;
34     END IF;
35     END IF;
36     SET dEVID = "BeLT";
37 ELSE
38     SET dEVID = SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'eventID=', -1), '',
39     1);
40 END IF;
41
42 INSERT INTO event (eCOUNT, SID, UID, TIME, MS, EVENTID, MSG)
43     VALUES(dECO, dSID, tUID, tTIME, c1, dEVID, tMSG);
44
45 IF tSDT LIKE "%mouse%" THEN
46     SET c1=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'X=', -1), '', 1);
47     SET c2=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'Y=', -1), '', 1);
48     IF INSTR(tSDT, 'delta=') = 0 THEN
49         SET c3=NULL;
50     ELSE
51         SET c3=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'delta=', -1), '', 1);
52     END IF;
53     INSERT INTO mouse (eCOUNT,SID,UID,xCoord,yCoord,wheelDelta)
54     VALUES(dECO, dSID, tUID, c1, c2, c3);
55 ELSEIF tSDT LIKE "%key%" THEN
56     SET c1=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'alt=', -1), '', 1);
57     SET t1=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'event=', -1), '', 1);
58     SET c2=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'count=', -1), '', 1);
59     INSERT INTO keyboard (eCOUNT, SID, UID, superKey, button, count)
60     VALUES(dECO, dSID, tUID, c1, t1, c2);
61 ELSE
62     IF tSDT LIKE "%belt%" THEN
63         SET t1=t2=NULL;
64     ELSE
65         SET t1=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'elemType=', -1), '',
66         1);
67         SET t2=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'elemDescription=', -1),
68         '', 1);
69     END IF;
70     IF INSTR(tSDT, 'rectBottomY=') = 0 THEN
71         SET c1=c2=c3=c4=NULL;
72     ELSE
73         SET c1=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'rectBottomY=', -1), '',
74         1);
75         SET c2=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'rectTopY=', -1), '',
76         1);
77         SET c3=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'rectLeftX=', -1), '',
78         1);
79         SET c4=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'rectRightX=', -1), '',
80         1);
81     END IF;
82
83     IF INSTR(tSDT, 'desc=') = 0 THEN
84         SET t3 = NULL;
85     ELSE
86         SET t3=SUBSTRING_INDEX(SUBSTRING_INDEX(tSDT, 'desc=', -1), '', 1);
87     END IF;

```

```

79
80     INSERT INTO automation(ECOUNT, SID, UID, PRCNAME, UIAID, elType, elDesc
      , descr, rBY,rTY,rLX,rRX)
81     VALUES(dECO, dSID, tUID, tPNM, tPID, t1, t2, t3, c1, c2, c3, c4);
82 END IF;
83
84 END //
85
86 DELIMITER ;

```

## D.5 Script to insert data into indexed database

### Code D.5: Indexed database script

```

1 DELIMITER ;
2 DROP DATABASE IF EXISTS syslog_indexed;
3
4 CREATE DATABASE syslog_indexed;
5 use syslog_indexed;
6
7 CREATE TABLE session (
8     SID INT(11) NOT NULL,
9     UID CHAR(35) NOT NULL,
10    START    TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
11    END CHAR(30) NULL
12 )ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
13
14 CREATE TABLE event (
15     ECOUNT INT(11) NOT NULL,
16     SID INT(11) NOT NULL,
17     UID CHAR(35) NOT NULL,
18     TIME    TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
19     MS INT(11),
20     EVENTID CHAR(6),
21     MSG TEXT
22 )ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
23
24 CREATE TABLE mouse (
25     ECOUNT INT(11),
26     SID INT(11),
27     UID CHAR(35),
28     xCoord INT(11),
29     yCoord INT(11),
30     wheelDelta CHAR(10)
31 )ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
32
33 CREATE TABLE keyboard (
34     ECOUNT INT(11),
35     SID INT(11),
36     UID CHAR(35),
37     superKey INT(10),
38     button VARCHAR(128),
39     count INT(11)
40 )ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
41
42 CREATE TABLE automation (
43     ECOUNT INT(11),
44     SID INT(11),
45     UID CHAR(35),
46     PRCNAME VARCHAR(261),
47     UIAID TEXT,
48     elType TEXT,
49     elDesc TEXT,
50     descr TEXT,
51     rBY INT(11),
52     rTY INT(11),
53     rLX INT(11),
54     rRx INT(11)

```

```

55 )ENGINE=MYISAM DEFAULT CHARACTER SET utf8;
56
57 CREATE INDEX seUID ON session(UID);
58 CREATE INDEX seSID ON session(SID);
59 CREATE INDEX seStart ON session(START);
60
61 CREATE INDEX evUID ON event(UID);
62 CREATE INDEX evSID ON event(SID);
63 CREATE INDEX evCount ON event(ECOUNT);
64 CREATE INDEX evEVID ON event(EVENTID);
65 CREATE INDEX evTime ON event(TIME);
66
67 CREATE INDEX keUID ON keyboard(UID);
68 CREATE INDEX keSID ON keyboard(SID);
69 CREATE INDEX keCount ON keyboard(ECOUNT);
70 CREATE INDEX keBut ON keyboard(button);
71
72 CREATE INDEX auUID ON automation(UID);
73 CREATE INDEX auSID ON automation(SID);
74 CREATE INDEX auCount ON automation(ECOUNT);
75 CREATE INDEX auPRCNAME ON automation (PRCNAME);
76
77 CREATE INDEX moUID ON mouse(UID);
78 CREATE INDEX moSID ON mouse(SID);
79 CREATE INDEX moCount ON mouse(ECOUNT);

```

## D.6 Bash script used to run the server test

Code D.6: Script at the server used for testing performance

```

1 #!/bin/bash
2
3 length=3600 # The length of each test in seconds
4 buffer=60 # Pause in between sessions
5 sarTimeBuffer=10 # Time before and after each session we start sar
6
7 # Total amount of time we run sar each session
8 totalSarTime=$((length + $sarTimeBuffer + $sarTimeBuffer + $buffer))
9
10 ps aux | grep syslog-mysql | awk '{print $2}' |
11 while read line;
12 do
13     kill $line
14 done
15
16
17 echo "Starting server with $length second intervals"
18 echo "Each test is open for $buffer seconds"
19 echo "Starting syslog-ng RAW server"
20 /etc/init.d/syslog-ng stop
21 cp ./raw.conf /etc/syslog-ng/syslog-ng.conf
22 /etc/init.d/syslog-ng start
23 sar -A -o time_raw.bin 1 $totalSarTime > /dev/null &
24 sleep $sarTimeBuffer
25 echo "Opening test for clients"
26 echo "1" > /var/www/belt_start_test.txt # Open test
27 sleep $buffer # Need to make sure everyone has started
28 echo "Closing test for clients, they should all be connected by now"
29 echo "0" > /var/www/belt_start_test.txt # Close test
30
31 sleep $sarTimeBuffer
32 sleep $length
33 echo "Starting syslog-ng CSV server"
34 /etc/init.d/syslog-ng stop
35 cp ./csv.conf /etc/syslog-ng/syslog-ng.conf
36 /etc/init.d/syslog-ng start
37 sar -A -o time_csv.bin 1 $totalSarTime > /dev/null &
38 sleep $sarTimeBuffer

```

```

39 echo "Opening test for clients"
40 echo "1" > /var/www/belt_start_test.txt # Open test
41 sleep $buffer # Need to make sure everyone has started
42 echo "Closing test for clients, they should all be connected by now"
43 echo "0" > /var/www/belt_start_test.txt # Close test
44
45
46 sleep $sarTimeBuffer
47 sleep $length
48 echo "Starting syslog-ng XML server"
49 /etc/init.d/syslog-ng stop
50 cp ./xml.conf /etc/syslog-ng/syslog-ng.conf
51 /etc/init.d/syslog-ng start
52 sar -A -o time_xml.bin 1 $totalSarTime > /dev/null &
53 sleep $sarTimeBuffer
54 echo "Opening test for clients"
55 echo "1" > /var/www/belt_start_test.txt # Open test
56 sleep $buffer # Need to make sure everyone has started
57 echo "Closing test for clients, they should all be connected by now"
58 echo "0" > /var/www/belt_start_test.txt # Close test
59
60
61 sleep $sarTimeBuffer
62 sleep $length
63 /etc/init.d/syslog-ng stop
64 /etc/syslog-ng/syslog-mysql_unindexed.sh &
65 cp ./db.conf /etc/syslog-ng/syslog-ng.conf
66 echo "Starting syslog-ng Database server"
67 /etc/init.d/syslog-ng start
68 sar -A -o time_db.bin 1 $totalSarTime > /dev/null &
69 sleep $sarTimeBuffer
70 echo "Opening test for clients"
71 echo "1" > /var/www/belt_start_test.txt # Open test
72 sleep $buffer # Need to make sure everyone has started
73 echo "Closing test for clients, they should all be connected by now"
74 echo "0" > /var/www/belt_start_test.txt # Close test
75
76 sleep $sarTimeBuffer
77 sleep $length
78 ps aux | grep syslog-mysql | awk '{print $2}' |
79 while read line;
80 do
81     kill $line
82 done
83
84 /etc/init.d/syslog-ng stop
85 /etc/syslog-ng/syslog-mysql_indexed.sh &
86 echo "Starting syslog-ng Database server"
87 /etc/init.d/syslog-ng start
88 sar -A -o time_db_i.bin 1 $totalSarTime > /dev/null &
89 sleep $sarTimeBuffer
90 echo "Opening test for clients"
91 echo "1" > /var/www/belt_start_test.txt # Open test
92 sleep $buffer # Need to make sure everyone has started
93 echo "Closing test for clients, they should all be connected by now"
94 echo "0" > /var/www/belt_start_test.txt # Close test

```

## D.7 RAW part of Syslog-NG configuration file

Code D.7: raw part of the syslog-ng.conf file

```

1 template raw_output {
2     template("${ISODATE}\",\"$MSGID\", \"$PROGRAM\", \"$PID\", \"$SDATA\", \"
    $MSGONLY\"\\n");
3 };

```

## D.8 CSV part of Syslog-NG configuration file

## Code D.8: CSV part of the syslog-ng.conf file

```

1 template csv_file {
2   template(
3     "${MSGID.NUM},$ISODATE,
4     $(if (\`${SDATA.UI@1.eventID}\` == \`\`)
5       $(if (\`${SDATA.key@1.eventID}\` == \`\`)
6         $(if (\`${SDATA.mouse@1.eventID}\` == \`\`)
7           \`${MSGONLY}\`
8           `${SDATA.mouse@1.eventID},${SDATA.mouse@1.X},${SDATA.mouse@1.Y}
9           $(if (\`${SDATA.mouse@1.delta}\` == \`\`)
10            \`\`
11            ,`${SDATA.mouse@1.delta}
12          )
13        )
14        `${SDATA.key@1.eventID},${SDATA.key@1.alt},${SDATA.key@1.event},
15        `${SDATA.key@1.count}
16      )
17      `${SDATA.UI@1.eventID},${SDATA.UI@1.elemType},${PID},${PROGRAM},
18      `${SDATA.UI@1.elemDescription}'
19      $(if (\`${SDATA.UI@1.desc}\` == \`\`)
20        $(if (\`${SDATA.UI@1.newvalue}\` == \`\`)
21          $(if (\`${SDATA.UI@1.rectBottomY}\` == \`\`)
22            \`\`
23            ,`${SDATA.UI@1.rectBottomY},${SDATA.UI@1.rectTopY},
24            `${SDATA.UI@1.rectLeftX},${SDATA.UI@1.rectRightX}
25          )
26          ,`${SDATA.UI@1.newvalue}'
27        )
28        ,`${SDATA.UI@1.desc}'
29      )
30    )
31    $(if (\`${SDATA.belt@1.event}\` == \`\`)
32      \`\`
33      `${SDATA.belt@1.event})\n");
34  };

```

## D.9 XML part of syslog-NG configuration file

## Code D.9: XML part of the syslog-ng.conf file

```

1 template xml_file {
2   template("
3     $(if (\`${SDATA.belt@1.event}\` != \`start\`)
4       \`\`
5       <events>
6     )
7     <event><num>`${MSGID.NUM}</num>
8     <date>`$ISODATE`</date>
9     $(if (\`${SDATA.UI@1.eventID}\` == \`\`)
10      $(if (\`${SDATA.key@1.eventID}\` == \`\`)
11        $(if (\`${SDATA.mouse@1.eventID}\` == \`\`)
12          \`\`
13          <id>`${SDATA.mouse@1.eventID}</id>
14          <x>`${SDATA.mouse@1.X}</x><y>`${SDATA.mouse@1.Y}</y>
15          $(if (\`${SDATA.mouse@1.delta}\` == \`\`)
16            \`\`
17            <delta>`${SDATA.mouse@1.delta}</delta>
18          )
19        )
20        <id>`${SDATA.key@1.eventID}</id>
21        <alt>`${SDATA.key@1.alt}</alt>
22        <key>`${SDATA.key@1.event}</key>
23        <count>`${SDATA.key@1.count}</count>
24      )
25      <id>`${SDATA.UI@1.eventID}</id>
26      <type>`${SDATA.UI@1.elemType}</type>
27      <uiid>`${PID}</uiid>

```

```

28 <program>${PROGRAM}</program>
29 <description>${.SDATA.UI@1.elemDescription}</description>
30 $(if (\`${.SDATA.UI@1.desc}\` == \`\`)
31     $(if (\`${.SDATA.UI@1.newvalue}\` == \`\`)
32         $(if (\`${.SDATA.UI@1.rectBottomY}\` == \`\`)
33             \`\`
34             <rectangle><bottomY>${.SDATA.UI@1.rectBottomY}</bottomY>
35             <topY>${.SDATA.UI@1.rectTopY}</topY>
36             <leftX>${.SDATA.UI@1.rectLeftX}</leftX>
37             <rightX>${.SDATA.UI@1.rectRightX}</rightX></rectangle>
38         )
39         <newValue>${.SDATA.UI@1.newvalue}</newValue>
40     )
41     <description2>${.SDATA.UI@1.desc}</description2>
42 )
43 )
44 $(if (\`${.SDATA.belt@1.event}\` == \`\`)
45     \`\`
46     <belt>${.SDATA.belt@1.event}</belt>
47 )
48 </event>
49 $(if (\`${.SDATA.belt@1.event}\` != \`stop\`)
50     \`\`
51     </events>)\n"
52 );
53 };

```

## D.10 Syslog-NG for database storage

Code D.10: Syslog-NG config for databse storage

```

1 destination d_sql2 {
2     pipe("/tmp/mysql.syslog-ng.pipe"
3         template("CALL log('${MSGID}', '${HOST}', '${ISODATE}', '${PROGRAM}', '${PID}',
4             '$SDATA', '$MSGONLY');\n") template-escape(yes));

```

## D.11 Bash script for inserting data into database

Code D.11: Bash script for inserting data to DB

```

1 #!/bin/bash
2 # Takes input from a FIFO list and executes the retrieved statement in the
3 # database. This then runs the decided procedure that inserts the data
4 # into the
5 # database.
6 # This is retrieved from a tutorial online at;
7 # http://chaos.untouchable.net/index.php/HOWTO_setup_syslog-
8 # ng_to_log_to_mysql
9 if [ -e /tmp/mysql.syslog-ng.pipe ]; then
10     while [ -e /tmp/mysql.syslog-ng.pipe ]; do
11         mysql syslog_indexed -u syslog --password=0x80sWaT < /tmp/mysql.syslog-
12             ng.pipe
13     done
14 else
15     mkfifo /tmp/mysql.syslog-ng.pipe
16 fi

```

# BeLT: Behavior Logging Tool

Robin Stenvi, Magnus Øverbø  
and Lasse T. Johansen

Generated with Doxygen 1.8.3

May 15, 2013

This product includes software developed by the OpenSSL Project  
for use in the OpenSSL Toolkit (<http://www.openssl.org/>)

# Contents

<b>1</b>	<b>Module Documentation</b>	<b>1</b>
1.1	List of all classes	1
1.2	Global variables	2
1.3	Misc Global Functions	3
1.4	All enumerations	5
1.5	Default Error Strings	6
1.6	Current server status at the client	7
1.7	Current mode of storage	8
1.8	Constants for success or failure messages	9
1.9	Which log event occurred	10
1.10	Events from user	11
1.11	Messages used throughout the application	12
1.12	Different colors used	13
1.13	Bif flags for active system keys	14
1.14	Each field in the update file	15
1.15	Each field in the list of patches	16
1.16	Diferent software events	17
1.17	Different colors for icons	18
1.18	All the global / private structs	19
1.19	All the key possible key variations	20
1.20	Global function for string manipulation	24
<b>2</b>	<b>Class Documentation</b>	<b>34</b>
2.1	AboutDialog Class Reference	34
2.2	Blacklist Struct Reference	35
2.3	Cbelt_mainApp Class Reference	35
2.4	Cbelt_mainDlg Class Reference	36
2.5	checkUpdate Class Reference	57
2.6	CTrayNot Class Reference	64
2.7	deviceInfo Struct Reference	66
2.8	eventHandler Class Reference	66
2.9	eventInfoUnion Struct Reference	72
2.10	Events Class Reference	73
2.11	sendData::Excluded Struct Reference	80
2.12	sendData::ExcludeIndex Struct Reference	80
2.13	sendData::File Struct Reference	80
2.14	filterSettings Class Reference	81
2.15	focusEventHandler Class Reference	87
2.16	formatData Class Reference	92
2.17	handleData Class Reference	95
2.18	HIDDevice Struct Reference	138
2.19	HWMonitor Class Reference	139
2.20	KeyboardDevice Struct Reference	141
2.21	KeyInfo Struct Reference	141
2.22	Keylogger Class Reference	142
2.23	keyType Struct Reference	146
2.24	handleData::!lastAll Struct Reference	146
2.25	Mouse Class Reference	147
2.26	MouseInfo Struct Reference	150
2.27	myWinEvent Class Reference	151
2.28	processList Struct Reference	153
2.29	sendData::progressRange Struct Reference	154
2.30	propertyEventHandler Class Reference	154
2.31	Screen Struct Reference	158
2.32	sendData Class Reference	159
2.33	SettingDialog Class Reference	173
2.34	Syslog1 Class Reference	183



---

2.35 sysResources Struct Reference . . . . .	192
2.36 sendData::Thread Struct Reference . . . . .	192
2.37 UIAutomation Class Reference . . . . .	193
<b>Index</b>	<b>195</b>

# 1 Module Documentation

## 1.1 List of all classes

All classes.

### Classes

- class [AboutDialog](#)  
*Sets the text and gets the version number from the registry.*
- class [Cbelt\\_mainApp](#)  
*Class that defines the application starting point, does not show a UI.*
- class [Cbelt\\_mainDlg](#)  
*Main dialog window that is displayed to the user.*
- class [checkUpdate](#)  
*Class that checks if a new update is available.*
- class [Events](#)  
*Retrieves UI properties that we send to the server.*
- class [eventHandler](#)  
*Responsible for registering for events and receiving these events.*
- class [filterSettings](#)  
*Handles all the user settings for filtering data to screen.*
- class [focusEventHandler](#)  
*Handles focus change events.*
- class [formatData](#)  
*Retrieves the real time that an event happened at and gives it as a readable string.*
- class [handleData](#)  
*In charge of writing all the data to the server, also does some filtering.*
- class [Keylogger](#)  
*Collect and organizes keyboard events before they are written to disk.*
- class [HWMonitor](#)  
*Class for monitoring Hardware usage.*
- class [Mouse](#)  
*Collect and organizes mouse events before they are written to disk.*
- class [propertyEventHandler](#)  
*Class that deals with all Property change events.*
- class [sendData](#)  
*Dialog that handles everything when user wants to send local file to server.*
- class [SettingDialog](#)  
*A class for setting certain config settings of BeLT.*
- class [Syslog1](#)  
*Handles all the connection to the server.*
- class [CTrayNot](#)  
*A class for creating/maintaining the system tray icon.*
- class [UIAutomation](#)  
*Handles the creation and destruction of all the UI Automation elements.*
- class [myWinEvent](#)  
*Implements MSAA functionality.*

### 1.1.1 Detailed Description

All classes.

## 1.2 Global variables

All the global variables used throughout the program.

### Variables

- [sendData](#) \* [senddata](#)  
*Dialog to send previously stored file to server.*
- [SettingDialog](#) \* [settingDlg](#)  
*creates the setting dialog object to be interacted with*
- [Keylogger](#) \* [keylogger](#)  
*Responsible for dealing with all keyboard input.*
- [Mouse](#) \* [mouselogger](#)  
*Responsible for dealing with all the mouse events.*
- [handleData](#) \* [Handledata](#)  
*Is the place where all events are sent, does some filtering and send it to the server.*
- [UIAutomation](#) \* [MyUiautomation](#)  
*Handles all Software events.*
- [Events](#) \* [Gevents](#)  
*Common functions for software events.*
- [myWinEvent](#) \* [winevents](#)  
*Handles MSAA events.*
- [HWMonitor](#) \* [HW](#)  
*Handles average system usage (CPU and memory)*
- HHOOK [keyhook](#)  
*Handle to our keyboard hook.*
- HHOOK [mousehook](#)  
*Handle to our mouse hook.*
- [Screen screens](#) [[MAX\\_SCREEN](#)S]  
*All the screens we have seen.*
- int [numScreens](#) = 0  
*How many screens we have seen so far.*
- const int [MAX\\_SCREEN](#)S = 20  
*Max number of screens we can hold.*

### 1.2.1 Detailed Description

All the global variables used throughout the program.

## 1.3 Misc Global Functions

Function for listening for keyboard events.

### Functions

- `__declspec (dllexport) LRESULT CALLBACK keyEvent(int code`  
*Callback function that is called anytime a key event occur.*
- `BOOL CALLBACK MyInfoEnumProc (HMONITOR hMonitor, HDC hdcMonitor, LPRECT lprcMonitor, LPARAM dwData)`  
*Callback function to enumerate all available monitors.*

### 1.3.1 Detailed Description

Function for listening for keyboard events.

1. Author Robin Stenvi Function for listening for mouse events Robin Stenvi

Miscellaneous global functions

### 1.3.2 Function Documentation

#### 1.3.2.1 `__declspec ( dllexport )`

Callback function that is called anytime a key event occur.

Callback function that is called anytime a mouse event occur.

1. Author Robin Stenvi

#### Parameters

<code>in</code>	<code>code</code>	Hook code (Supplied by Windows)
<code>in</code>	<code>wParam</code>	Metadata about the event (Supplied by Windows)
<code>in</code>	<code>lParam</code>	KBDLLHOOKSTRUCT* (Supplied by Windows)

#### Remarks

This should really be in a separate DLL, but we don't have to, so we don't do it. If you need a high level hook, you have to have it in a separate DLL.

1. Author Robin Stenvi

#### Parameters

<code>in</code>	<code>code</code>	Hook code (Supplied by Windows)
<code>in</code>	<code>wParam</code>	Metadata about the event (Supplied by Windows)
<code>in</code>	<code>lParam</code>	MSLLHOOKSTRUCT* (Supplied by Windows)

#### Remarks

This should really be in a separate DLL, but we don't have to, so we don't do it. If you need a high level hook, you have to have it in a separate DLL.

### 1.3.2.2 BOOL CALLBACK MyInfoEnumProc ( HMONITOR *hMonitor*, HDC *hdcMonitor*, LPRECT *lprcMonitor*, LPARAM *dwData* )

Callback function to enumerate all available monitors.

If we find a new monitor we will add it to the list. This function can be called at any time. If we see a monitor we have stored before, we will ignore it.

1. Author Robin Stenvi

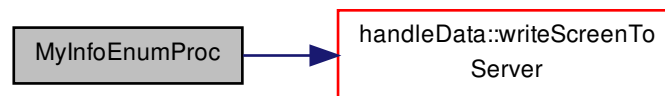
#### Parameters

in	<i>hMonitor</i>	Physical display monitor
in	<i>hdcMonitor</i>	Device context for monitor
in	<i>lprcMonitor</i>	Rectangle for the monitor
in	<i>dwData</i>	Unused application-defined data

#### Returns

Returns FALSE if we don't have space for more screens, otherwise it returns TRUE.

Here is the call graph for this function:



## 1.4 All enumerations

All enumerations, both globally defined and private.

### Enumerations

- enum [Level](#)  
*Describes the level of the syslog message.*
- enum [Facility](#)  
*Describes the facility level of the syslog message.*
- enum [ERRORS](#)  
*Enum to describe if something went wrong when trying to update.*
- enum [unionType](#)  
*Enum to see which union is used in eventInfoUnion.*

#### 1.4.1 Detailed Description

All enumerations, both globally defined and private.

#### 1.4.2 Enumeration Type Documentation

##### 1.4.2.1 enum [Level](#)

Describes the level of the syslog message.

Can be used to set a priority level, we set all messages to be information

##### 1.4.2.2 enum [Facility](#)

Describes the facility level of the syslog message.

Can be used to increase priority level, we set all messages to be Local0

##### 1.4.2.3 enum [ERRORS](#)

Enum to describe if something went wrong when trying to update.

The calling function should use this to indicate what went wrong and maybe take action, some will indicate server error while other indicate that the user aborted, while other again might indicate that something is wrong in the code.

## 1.5 Default Error Strings

Some standards for error messages that are sent as messages.

### Macros

- `#define ErrorTime (std::string)"1970-01-01T00:00:00.0000Z"`  
*Timestamp that can be used in a syslog message.*
- `#define ErrorClock (std::string)"00:00:00"`  
*Just the time without the date, not valid in syslog message.*
- `#define ErrorDate (std::string)"1970-01-01 00:00:00"`  
*Date and time used as filename.*
- `#define ErrorSD (std::string)""`  
*Structured data can be empty, server will still print an error message in their file.*
- `#define ErrorCsv (std::string)"error\n"`  
*Defines the entire CSV line, only contains one element, but is valid CSV.*
- `#define ErrorCsvRectangle (std::string)"-1,-1,-1,-1"`  
*Used as empty rectangle, or error when creating the string, used in CSV files.*
- `#define ErrorFormatRectangle (std::string)"bottY=\"-1\" topY=\"-1\" leftX=\"-1\" rightX=\"-1\""`  
*Empty Rectangle or error when creating rectangle, used in syslog string.*
- `#define ErrorServer (std::string)"<134>1 1970-01-01T00:00:00.0000Z error error error 0_0\n"`  
*Entire syslog message, in a valid format.*
- `#define ErrorMouseSent (std::string)"Unknown mouse event"`  
*Error message for descriptive mouse sentence to the user.*
- `#define ErrorKeySent (std::string)"Unknown key event"`  
*Error message for descriptive key event sentence to the user.*
- `#define replaceInvalid 'X'`  
*What we replace invalid characters with, in the Syslog protocol.*
- `#define UnknownElem "|unknown|"`  
*Whenever we fail to retrieve information about a software element.*
- `#define TooLongElem (std::string)"|Too long|"`  
*Whenever the text in an element is longer than some value.*
- `#define EmptyElem (std::string)"|empty|"`  
*If the text we are trying to retrieve is empty.*

### 1.5.1 Detailed Description

Some standards for error messages that are sent as messages. These messages should be safe to send in a syslog message or printed in a CSV file.

## 1.6 Current server status at the client

This defines the current status, regarding the connection with the server.

### Macros

- #define `SERVER_STOPPED` 0  
*Server does NOT have an active connection with the server.*
- #define `SERVER_PAUSED` 1  
*Connection with server is paused, underlying connection is there.*
- #define `SERVER_RUNNING` 2  
*We have an active connection with the server.*

### 1.6.1 Detailed Description

This defines the current status, regarding the connection with the server. These are all the possible states and it can only be one of them.



## 1.7 Current mode of storage

This defines the current status, regarding how logs are stored, only one of the modes is possible.

### Macros

- #define `STORAGE_SERVER` 1  
*We are sending concurrently to the server.*
- #define `STORAGE_LOCAL` 2  
*We store in local raw files before sending to the server.*
- #define `STORAGE_CSV` 3  
*We only store CSV files.*

### 1.7.1 Detailed Description

This defines the current status, regarding how logs are stored, only one of the modes is possible.

## 1.8 Constants for success or failure messages

Constants to define messages of success or fail, so the GUI know what has happened.

### Macros

- #define `SSLFAIL` 1  
*We have failed to initialize SSL/TLS.*
- #define `SSLSUCCESS` 2  
*We have succeeded to initialize SSL/TLS.*
- #define `SSLSUCCESSPAUSE` 3  
*We have successfully paused SSL/TLS.*

### 1.8.1 Detailed Description

Constants to define messages of success or fail, so the GUI know what has happened.

## 1.9 Which log event occurred

Defines to define what type of event has happened, each uses one bit, because they are power of two.

### Macros

- #define UIEVENT 1  
*Software event.*
- #define KEYEVENT 2  
*All key events.*
- #define MOUSEEVENT 4  
*All mouse events.*
- #define KEYUP 8  
*Only key released.*
- #define KEYDOWN 16  
*Only key pressed.*
- #define MOUSEMOVE 32  
*All mouse movements.*
- #define MOUSEPRESS 64  
*All mouse up and down buttons.*
- #define MOUSEWHEEL 128  
*Any mouse wheel action.*
- #define MOUSEUP 256  
*All mouse buttons released.*
- #define MOUSEDOWN 512  
*All mouse buttons pressed.*
- #define MOUSELEFT 1024  
*Pressed and released for left mouse button.*
- #define MUSERIGHT 2048  
*Pressed and released for right mouse button.*
- #define MOUSEMIDDLE 4096  
*Pressed and released for middle mouse button.*
- #define MOUSEWHEEL2 8192  
*Mouse wheel button.*
- #define MOUSEMAX 8192  
*Defines the max bit for mouse events.*

### 1.9.1 Detailed Description

Defines to define what type of event has happened, each uses one bit, because they are power of two.

## 1.10 Events from user

Used to determine the event that happened in `handleData`.

### Macros

- `#define LISTENER_START 1`  
*Start new session.*
- `#define LISTENER_STOP 2`  
*Stop current session.*
- `#define LISTENER_PAUSE 3`  
*Pause connection.*
- `#define LISTENER_RESUME 4`  
*Resume from paused state.*

### 1.10.1 Detailed Description

Used to determine the event that happened in `handleData`.

## 1.11 Messages used throughout the application

Our own defined message variables.

### Macros

- #define `WM_PRINTEVENT` (WM\_USER + 850)  
*Event that should be printed to screen.*
- #define `WM_NOTIFY_PAUSE` (WM\_USER + 900)  
*The user has pressed shortcut for pause.*
- #define `WM_NOTIFY_FAIL` (WM\_USER + 950)  
*Something has failed.*
- #define `WM_NOTIFY_SUCCESS` (WM\_USER + 951)  
*Something has succeeded.*
- #define `WM_KEYSTROKE_MESSAGE` (WM\_USER + 970)  
*Not used.*
- #define `WM_KEYSTROKE_SHUTDOWN_MESSAGE` (WM\_USER + 971)  
*Not used.*
- #define `WM_MOUSE_MESSAGE` (WM\_USER + 972)  
*Not used.*
- #define `WM_MOUSE_SHUTDOWN_MESSAGE` (WM\_USER + 973)  
*Not used.*

### 1.11.1 Detailed Description

Our own defined message variables.

## 1.12 Different colors used.

Different colors used throughout the program.

### Macros

- #define `COLOR_BLACK` RGB(0,0,0)  
*Used to print information messages, like start/stop.*
- #define `COLOR_GREEN` RGB(0,255,0)  
*Used to print success messages.*
- #define `COLOR_DGREEN` RGB(0,128,0)  
*Keyboard messages.*
- #define `COLOR_BLUE` RGB(0,0,255)  
*Used to print mouse messages.*
- #define `COLOR_RED` RGB(255,0,0)  
*Used to print failed messages.*

### 1.12.1 Detailed Description

Different colors used throughout the program.

## 1.13 Bif flags for active system keys

Bit-flags to determine function keys on the keyboard, gives context to key pressed.

### Macros

- #define `MYALT_PRESSED` 1  
*Left or right alt key.*
- #define `MYCTRL_PRESSED` 2  
*Left or right ctrl.*
- #define `MYSHIFT_PRESSED` 4  
*Left or right shift.*
- #define `MYWINDOWS_PRESSED` 8  
*Left or right Windows key.*
- #define `CAPS_LOCK_ACTIVE` 16  
*Caps lock is active, not necessarily pressed.*
- #define `NUM_LOCK_ACTIVE` 32  
*Num lock is active, not necessarily pressed.*
- #define `SCROLL_LOCK_ACTIVE` 64  
*Scroll lock is active, not necessarily pressed.*

### 1.13.1 Detailed Description

Bit-flags to determine function keys on the keyboard, gives context to key pressed. Several can be set

## 1.14 Each field in the update file

Defines the order of the field in the update configuration and the number of min fields.

### Macros

- #define `UPDATE_IP` 0  
*IP or URL to server for update files.*
- #define `UPDATE_PORT` 1  
*The port we should use for update (http)*
- #define `UPDATE_PORT_TLS` 2  
*The port we should use for TLS communication.*
- #define `UPDATE_PATH` 3  
*The path at the server for update configuration file.*
- #define `VERSION_NUM` 4  
*Newest version number.*
- #define `PATCH_LIST_PATH` 5  
*Full path at the server to file of patch list.*
- #define `LOG_IP` 6  
*IP or URL to logging server.*
- #define `LOG_PORT` 7  
*The port we should send to when logging.*
- #define `MIN_UPDATE_CONFIG` 7  
*The minimum number of fields we need for this to be a valid config file.*

### 1.14.1 Detailed Description

Defines the order of the field in the update configuration and the number of min fields.



## 1.15 Each field in the list of patches

Defines the order of the field in the patch list and the number of fields.

### Macros

- #define `DEP_VERSION` 0  
*Version number needed to use this patch.*
- #define `PLATFORM_32` 1  
*Patch for 32-bit architecture.*
- #define `PLATFORM_64` 2  
*Patch for 64-bit architecture.*
- #define `MIN_PATCH_LIST` 3  
*The minimum number of fields we need for this to be a valid config file.*

### 1.15.1 Detailed Description

Defines the order of the field in the patch list and the number of fields.

## 1.16 Diferent software events

List of flags for UI events we gather, used for faster processing later on when we filter the data.

### Macros

- #define **FC** 1  
*Focus change.*
- #define **OCS** 2  
*Object change state.*
- #define **VC** 3  
*Visual change.*
- #define **WO** 4  
*Window opened.*
- #define **EI** 5  
*Element invoked.*
- #define **MO** 6  
*Menu opened.*
- #define **TC** 7  
*Text changed.*
- #define **MMS** 8  
*Menu mode started.*

### 1.16.1 Detailed Description

List of flags for UI events we gather, used for faster processing later on when we filter the data.

## 1.17 Different colors for icons

All the possible colors for the icon.

### Macros

- #define RED\_ICON 0  
*Red icon used, when logging has stopped.*
- #define GREEN\_ICON 1  
*Green icon used, when logging has started.*
- #define BLUE\_ICON 2  
*Blue icon used, when logging has paused.*
- #define YELLOW\_ICON 3  
*Yellow icon used, when we have detected a password field.*

### 1.17.1 Detailed Description

All the possible colors for the icon.

## 1.18 All the global / private structs

All the global and private structs that are used throughout the program.

### Classes

- struct [sysResources](#)  
*Contains information about system resources in use by the system, all values should be in percentage.*
- struct [deviceInfo](#)  
*Information about a storage device, only a timestamp and a value that says whether it was inserted or removed.*
- struct [HIDDevice](#)  
*Information about an input device.*
- struct [KeyboardDevice](#)  
*Contains information about a keyboard device.*
- struct [eventInfoUnion](#)  
*Contains the information we store about each UI event.*
- struct [processList](#)  
*A list of processes so we can retrieve that information faster.*
- struct [Blacklist](#)  
*Whenever we see a value we are unable to retrieve we can use this to maybe save some time.*
- struct [Screen](#)  
*Information about a physical screen.*
- struct [KeyInfo](#)  
*All variable info we need to write about a key event.*
- struct [MouseInfo](#)  
*All variable info we need to write about a mouse event.*
- struct [handleData::lastAll](#)  
*Holds all the previous events, is used to find which events correlate to other events.*
- struct [keyType](#)  
*Struct to hold the number of key down we receive for some key.*

### 1.18.1 Detailed Description

All the global and private structs that are used throughout the program.

## 1.19 All the key possible key variations

These are all the possible key values that can be printed, except for those ASCII values that doesn't contain whitespace.

### Macros

- #define `KEYBOARDBACK` "|backspace|"  
*Backspace.*
- #define `KEYBOARDSPACE` "|space|"  
*Spacebar.*
- #define `KEYBOARDSPACE` "|space|"  
*Spacebar.*
- #define `KEYBOARDTAB` "|tab|"  
*Tab.*
- #define `KEYBOARDCLEAR` "|clear|"  
*Clear or form feed.*
- #define `KEYBOARDENTER` "|enter|"  
*Enter.*
- #define `KEYBOARDSHIFT` "|shift|"  
*Any shift, usually they are left or right.*
- #define `KEYBOARDCTRL` "|ctrl|"  
*Any control key, usually they are right or left.*
- #define `KEYBOARDALT` "|alt|"  
*Any alt key, usually they are right or left.*
- #define `KEYBOARDPAUSE` "|pause|"  
*Pause key.*
- #define `KEYBOARDCAPSLOCK` "|capsLock|"  
*Caps lock key.*
- #define `KEYBOARDESC` "|esc|"  
*Escape.*
- #define `KEYBOARDPGUP` "|PgUp|"  
*Page up.*
- #define `KEYBOARDPGDOWN` "|PgDn|"  
*Page Down.*
- #define `KEYBOARDEND` "|end|"  
*End.*
- #define `KEYBOARDHOME` "|home|"  
*Home.*
- #define `KEYBOARDLEFT` "|left|"  
*Left arrow.*
- #define `KEYBOARDUP` "|up|"  
*Up arrow.*
- #define `KEYBOARDRIGHT` "|right|"  
*Right arrow.*
- #define `KEYBOARDDOWN` "|down|"  
*Down arrow.*
- #define `KEYBOARDSELECT` "|select|"  
*Select key.*
- #define `KEYBOARDPRINT` "|print|"  
*Print key, don't think they exist today.*

- #define `KEYBOARDEXEC` `"|exec|"`  
*Execute, don't think they exist today.*
- #define `KEYBOARDPRTSC` `"|PrtSc|"`  
*Print screen.*
- #define `KEYBOARDINSERT` `"|insert|"`  
*Insert key.*
- #define `KEYBOARDDEL` `"|del|"`  
*Delete.*
- #define `KEYBOARDHELP` `"|help|"`  
*Help key.*
- #define `KEYBOARDLWINDOWS` `"|LWindows|"`  
*Left Windows.*
- #define `KEYBOARDRWINDOWS` `"|RWindows|"`  
*Right Windows.*
- #define `KEYBOARDAPPKEY` `"|appKey|"`  
*Application key on some Microsoft keyboard.*
- #define `KEYBOARDSLEEP` `"|sleep|"`  
*Computer sleep key.*
- #define `KEYBOARDNX` `"NX"`  
*Numbers for num-pad, X is a value between 0 and 9, I don't think we usually receive these events, they just come in as 0-9.*
- #define `KEYBOARDNTIMES` `"N*"`  
*\* on the num-pad (I think)*
- #define `KEYBOARDNPLUS` `"N+"`  
*+ on the num-pad (I think)*
- #define `KEYBOARDNDECIMAL` `"N,"`  
*, on the num-pad (I think)*
- #define `KEYBOARDNSUBTRACT` `"N-"`  
*- on the num-pad (I think)*
- #define `KEYBOARDNPERIOD` `"N."`  
*Period on the num-pad (I think)*
- #define `KEYBOARDNDIVIDE` `"N/"`  
*/ on the num-pad (I think)*
- #define `KEYBOARDFKEY` `"FX"`  
*All the F keys, X is replaced with a number from 1 to 24.*
- #define `KEYBOARDNUMLOCK` `"|NumLK|"`  
*Num lock.*
- #define `KEYBOARDSCROLLLOCK` `"|ScrLK|"`  
*Scroll lock.*
- #define `KEYBOARDLSHIFT` `"|Lshift|"`  
*Left shift.*
- #define `KEYBOARDRSHIFT` `"|Rshift|"`  
*Right shift.*
- #define `KEYBOARDLCTRL` `"|Lctrl|"`  
*Left control.*
- #define `KEYBOARDRCTRL` `"|Rctrl|"`  
*Right control.*
- #define `KEYBOARDLALT` `"|Lalt|"`  
*Left alt.*
- #define `KEYBOARDRALT` `"|Ralt|"`  
*Right alt.*

- #define `KEYBOARDROWSERBACK` "|Bback|"   
 *Browser back (special hardware)*
- #define `KEYBOARDROWSERFORWARD` "|Bforward|"   
 *Browser forward (special hardware)*
- #define `KEYBOARDROWSERREFRESH` "|Brefresh|"   
 *Browser refresh (special hardware)*
- #define `KEYBOARDROWSERSTOP` "|Bstop|"   
 *Browser stop (special hardware)*
- #define `KEYBOARDROWSERSEARCH` "|Bsearch|"   
 *Browser search (special hardware)*
- #define `KEYBOARDROWSERFAVORITES` "|Bfavorites|"   
 *Browser favorites (special hardware)*
- #define `KEYBOARDROWSERHOME` "|Bhome|"   
 *Browser home (special hardware)*
- #define `KEYBOARDVMUTE` "|Vmute|"   
 *Volume mute (special hardware)*
- #define `KEYBOARDVDOWN` "|Vdown|"   
 *Volume down (special hardware)*
- #define `KEYBOARDVUP` "|Vup|"   
 *Volume up (special hardware)*
- #define `KEYBOARDMNEXT` "|Mnext|"   
 *Multimedia next (special hardware)*
- #define `KEYBOARDMPREV` "|Mprev|"   
 *Multimedia previous (special hardware)*
- #define `KEYBOARDMSTOP` "|Mstop|"   
 *Multimedia stop (special hardware)*
- #define `KEYBOARDMPAUSE` "|Mpause|"   
 *Multimedia pause / play (special hardware)*
- #define `KEYBOARDLMAIL` "|Lmail|"   
 *Launch mail (special hardware)*
- #define `KEYBOARDLSELECT` "|Lselect|"   
 *Launch media select (special hardware)*
- #define `KEYBOARDLAPP1` "|Lapp1|"   
 *Launch application 1 (special hardware)*
- #define `KEYBOARDLAPP2` "|Lapp2|"   
 *Launch application 2 (special hardware)*
- #define `KEYBOARDOEM1` "|OEM1|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM2` "|OEM2|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM3` "|OEM3|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM4` "|OEM4|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM5` "|OEM5|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM6` "|OEM6|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM7` "|OEM7|"   
 *Vary by region (should not happen)*
- #define `KEYBOARDOEM8` "|OEM8|"   
 *Vary by region (should not happen)*

- Vary by region (should not happen)*

  - #define `KEYBOARDOEM102` "|OEM102|"
- Vary by region (should not happen)*

  - #define `KEYBOARDUNICODE` "|unicode|"

*Used to pass unicode characters, don't think this can happen from keyboard.*
- #define `KEYBOARDATTN` "|attn|"

*Attention key (older keyboards)*
- #define `KEYBOARDCRSEL` "|crsel|"

*Not sure what it does, but from the past.*
- #define `KEYBOARDEXSEL` "|exsel|"

*Not sure what it does, but from the past.*
- #define `KEYBOARDEROF` "|erof|"

*Erase end-of-file.*
- #define `KEYBOARDPLAY` "|play|"

*Play key.*
- #define `KEYBOARDZOOM` "|zoom|"

*Zoom key.*
- #define `KEYBOARDPA1` "|pa1|"

*Attention key.*
- #define `KEYBOARDCLEAR2` "|clear2|"

*Think it's the same as the first clear.*
- #define `KEYBOARDPASSWORD` "|\*|"

*What is used when we discover a password field.*
- #define `KEYBOARDUNKNOWN` "|unknown|";

*What we use if we don't manage to find the correct key.*

### 1.19.1 Detailed Description

These are all the possible key values that can be printed, except for those ASCII values that doesn't contain whitespace. For a full list of all possible key-codes, see [http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx)

### 1.19.2 Macro Definition Documentation

#### 1.19.2.1 #define KEYBOARDNX "NX"

Numbers for num-pad, X is a value between 0 and 9, I don't think we usually receive these events, they just come in as 0-9.

#### 1.19.2.2 #define KEYBOARDFKEY "FX"

All the F keys, X is replaced with a number from 1 to 24.



## 1.20 Global function for string manipulation

Global functions for manipulating strings, both conversions and change / find of the string.

### Functions

- `std::string` [replaceLetter](#) (`std::string &data`, `const char *from`, `const char *to`)  
*Replaces all character sequences in a string with another character sequence.*
- `bool` [intToString](#) (`int value`, `std::string &str`)  
*Converts an integer to `std::string`.*
- `bool` [dwordToString](#) (`DWORD value`, `std::string &str`)  
*Converts an unsigned integer to `std::string`.*
- `void` [cstringToString](#) (`CString s`, `std::string &str`)  
*Convert a `CString` to `std::string`.*
- `bool` [bstrToString](#) (`const wchar_t *pstr`, `long wslen`, `std::string &str`)  
*Converts a `bstr` to `std::string`.*
- `void` [fromBstrToCString](#) (`BSTR s`, `CString &str`)  
*Converts `BSTR` to `CString`.*
- `template<class T >`  
`std::string` [valueToString](#) (`T value`, `std::string &str`)  
*Takes any value and formats it as `std::string`.*
- `bool` [myAtoi](#) (`const std::string str`, `int &value`)  
*atoi function that checks if all the characters are numbers*
- `std::string` [getBetweenChar](#) (`const std::string data`, `const char from`, `const char to`)  
*Returns the string between two characters.*
- `void` [escapeLetter](#) (`std::string &data`, `const char *from`, `const char *to`)  
*Escape a string or a single letter in a `std::string`.*
- `std::vector< std::wstring >` [explode](#) (`const std::wstring &str`, `const wchar_t &ch`)  
*Takes in an `std::wstring` and returns a vector where "ch" is the delimiter.*
- `std::vector< std::string >` [explode](#) (`const std::string &str`, `const char &ch`)  
*Takes in an `std::string` and returns a vector where "ch" is the delimiter.*
- `std::wstring` [getBetweenWChar](#) (`const std::wstring data`, `const wchar_t from`, `const wchar_t to`, `bool last=false`)  
*Retrieves the all the characters between two characters, excluding those characters.*
- `SYSTEMTIME` [getTimeFromString](#) (`std::wstring str`, `const wchar_t dateSep= '-'`, `const wchar_t sep= ' '`, `const wchar_t timeSep= ':'`)  
*Converts a string to a `SYSTEMTIME` structure.*
- `std::wstring` [s2ws](#) (`const std::string &s`)  
*Converts `std::string` to `std::wstring`.*
- `std::string` [ws2s](#) (`const std::wstring &s`)  
*Converts `std::wstring` to `std::string`.*
- `int` [getProfileInfoInt](#) (`bool user`, `std::wstring section`, `std::wstring key`)  
*Wrapper to `getProfileInfo` that you can use if you need to retrieve an integer.*
- `DWORD` [writeProfileInfoInt](#) (`bool user`, `std::wstring section`, `std::wstring key`, `int data`)  
*Wrapper to `writeProfileInfo` that you can use if you need to write an integer.*
- `std::wstring` [getProfileInfo](#) (`bool user`, `std::wstring section`, `std::wstring key`)  
*Returns a variable in one of the applications program files.*
- `DWORD` [writeProfileInfo](#) (`bool user`, `std::wstring section`, `std::wstring key`, `std::wstring data`)  
*Writes a string in one of the applications program files.*
- `std::string` [generateRandomString](#) (`std::string alphabet`, `int length`)  
*Generates a random string.*

- bool [openFile](#) (GUID dir, std::wstring dir2, std::wstring name, CStdioFile &file, std::wstring &out-Name, UINT permissions=CFile::modeCreate|CFile::modeWrite|CFile::modeRead)  
*Opens a file in a given Windows directory.*
- std::string [printUSASCII](#) (std::string &str)  
*Make sures that the string is validated, in USASCII, according the the Syslog protocol.*
- std::string [base64Encode](#) (std::vector< BYTE > inputBuffer)  
*Converts a string to base 64, without padding.*

### 1.20.1 Detailed Description

Global functions for manipulating strings, both conversions and change / find of the string.

### 1.20.2 Function Documentation

#### 1.20.2.1 std::string replaceLetter ( std::string & data, const char \* from, const char \* to )

Replaces all character sequences in a string with another character sequence.

1. Author Robin Stenvi

#### Parameters

in, out	<i>data</i>	The string that should be modified, is modified
in	<i>from</i>	The sequence that should be explained
in	<i>to</i>	What from should be replaced with

#### Returns

Returns the modified string

#### 1.20.2.2 bool intToString ( int value, std::string & str )

Converts an integer to std::string.

1. Author Robin Stenvi

#### Parameters

in	<i>value</i>	The int that should be converted
out	<i>str</i>	The string that is returned

#### Returns

Returns true if we succeed, or false if we fail.

#### 1.20.2.3 bool dwordToString ( DWORD value, std::string & str )

Converts an unsigned integer to std::string.

1. Author Robin Stenvi

**Parameters**

in	<i>value</i>	The unsigned int that should be converted
out	<i>str</i>	The string that is returned

**Returns**

Returns true if we succeed, or false if we fail.

**1.20.2.4 void cstringToString ( CString s, std::string & str )**

Convert a CString to std::string.

1. Author Robin Stenvi

**Parameters**

in	<i>s</i>	The string that should be converted
out	<i>str</i>	s converted to std::string

**1.20.2.5 bool bstrToString ( const wchar\_t \* pstr, long wslen, std::string & str )**

Converts a bstr to std::string.

1. Author Robin Stenvi

**Parameters**

in	<i>pstr</i>	The string that should be converted
in	<i>wslen</i>	The length of the previous array, in characters, alternatively it can be -1 if pstr is NULL-terminated.
out	<i>str</i>	The output string

**Returns**

Returns false if we fail or true if we succeed

**1.20.2.6 void fromBstrToCString ( BSTR s, CString & str )**

Converts BSTR to CString.

1. Author Robin Stenvi

**Parameters**

in	<i>s</i>	The string that should be converted
out	<i>str</i>	The string that is returned

**1.20.2.7 template<class T > std::string valueToString ( T value, std::string & str )**

Takes any value and formats it as std::string.

For a list of supported values see here: <http://www.cplusplus.com/reference/ostream/ostream/operator%3C%3C/> It does no error checking, so it is up to the caller to check if it was successful. It is very inefficient so should be avoided when possible. Need to be defined in the header file because it's a template

function.

#### Parameters

<i>in</i>	<i>value</i>	The value that should be converted.
<i>out</i>	<i>str</i>	The string that contains the output when we are finished.

#### Returns

Returns the same as *str*.

1. Author Robin Stenvi

#### 1.20.2.8 bool myAtoi ( const std::string *str*, int & *value* )

atoi function that checks if all the characters are numbers

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>str</i>	The string that should be a number
<i>out</i>	<i>value</i>	The int, it contains the same as atoi if the return value is false

#### Returns

Returns false if the string is not a valid number, true if it is.

#### Remarks

Does not take into account integer overflow

#### 1.20.2.9 std::string getBetweenChar ( const std::string *data*, const char *from*, const char *to* )

Returns the string between two characters.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>data</i>	The whole string
<i>in</i>	<i>from</i>	The first character we are looking for
<i>in</i>	<i>to</i>	The second character we are looking for

#### Returns

Returns a string from the character after *from* and before *to*

#### 1.20.2.10 void escapeLetter ( std::string & *data*, const char \* *from*, const char \* *to* )

Escape a string or a single letter in a std::string.

1. Author Robin Stenvi

#### Parameters

<i>in, out</i>	<i>data</i>	The string that should be escaped, is updated.
<i>in</i>	<i>from</i>	The value to escape
<i>in</i>	<i>to</i>	What to escape the value with

#### Returns

The new `std::string` with escaped characters, this contains the same as `data`.

#### 1.20.2.11 `std::vector<std::wstring> explode ( const std::wstring & str, const wchar_t & ch )`

Takes in an `std::wstring` and returns a vector where "ch" is the delimiter.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>str</i>	The string we want to split.
<i>in</i>	<i>ch</i>	The delimiter

#### Returns

A vector that contains all the values.

#### 1.20.2.12 `std::vector<std::string> explode ( const std::string & str, const char & ch )`

Takes in an `std::string` and returns a vector where "ch" is the delimiter.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>str</i>	The string that should be exploded
<i>in</i>	<i>ch</i>	The delimiter

#### Returns

Returns a vector with each string

#### 1.20.2.13 `std::wstring getBetweenWChar ( const std::wstring data, const wchar_t from, const wchar_t to, bool last )`

Retrieves the all the characters between two characters, excluding those characters.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>data</i>	The whole string
<i>in</i>	<i>from</i>	The character we are going to start at
<i>in</i>	<i>to</i>	The character we are going to stop at
<i>in</i>	<i>last</i>	If true, we will look for the last occurrence of to and from

**Returns**

Returns an `std::wstring` with all the characters, returns empty string if we fail

#### 1.20.2.14 SYSTEMTIME `getTimeFromString ( std::wstring str, const wchar_t dateSep, const wchar_t sep, const wchar_t timeSep )`

Converts a string to a SYSTEMTIME structure.

1. Author Robin Stenvi

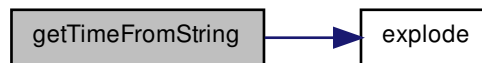
**Parameters**

<i>in</i>	<i>str</i>	The string containing the date and time
<i>in</i>	<i>dateSep</i>	The character that separates the data numbers
<i>in</i>	<i>sep</i>	The character that separates the date and the time
<i>in</i>	<i>timeSep</i>	The character that seperates each time value

**Returns**

A SYSTEMTIME structure representing the time, the string represent. Returns all 0's on failure.

Here is the call graph for this function:



#### 1.20.2.15 `std::wstring s2ws ( const std::string & s )`

Converts `std::string` to `std::wstring`.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>s</i>	The string that should be converted
-----------	----------	-------------------------------------

**Returns**

A `std::wstring` converted, is empty if we fail

#### 1.20.2.16 `std::string ws2s ( const std::wstring & s )`

Converts `std::wstring` to `std::string`.

1. Author Robin Stenvi

#### Parameters

<code>in</code>	<code>s</code>	The string that should be converted
-----------------	----------------	-------------------------------------

#### Returns

A `std::string` converted

1.20.2.17 `int getProfileInfoInt ( bool user, std::wstring section, std::wstring key )`

Wrapper to `getProfileInfo` that you can use if you need to retrieve an integer.

1. Author Robin Stenvi

#### Parameters

<code>in</code>	<code>user</code>	Says whether we need to get data from the user file or global file
<code>in</code>	<code>section</code>	Which section we are going to look in
<code>in</code>	<code>key</code>	What the key name is

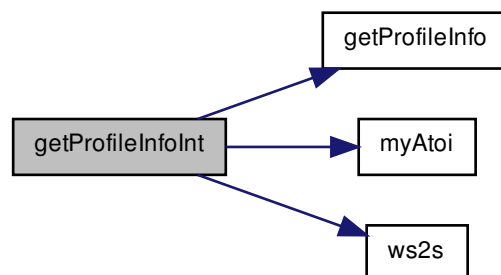
#### Returns

Returns the variable or -1 on failure.

#### Remarks

Checks if the variable is a proper integer, if not it will return -1.

Here is the call graph for this function:



1.20.2.18 `DWORD writeProfileInfoInt ( bool user, std::wstring section, std::wstring key, int data )`

Wrapper to `writeProfileInfo` that you can use if you need to write an integer.

1. Author Robin Stenvi

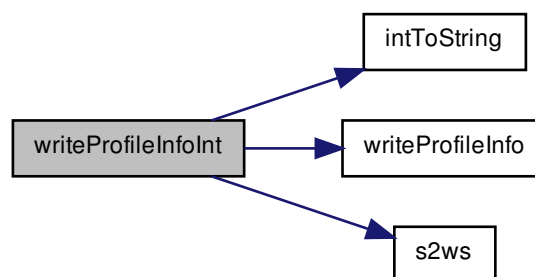
#### Parameters

<i>in</i>	<i>user</i>	Says whether we need to get data from the user file or global file
<i>in</i>	<i>section</i>	Which section we are going to look in
<i>in</i>	<i>key</i>	What the key name is
<i>in</i>	<i>data</i>	The integer that should be written

#### Returns

Returns 0 on success, otherwise -1 or GetLastError().

Here is the call graph for this function:



#### 1.20.2.19 `std::wstring getProfileInfo ( bool user, std::wstring section, std::wstring key )`

Returns a variable in one of the applications program files.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>user</i>	Says whether we need to get data from the user file or global file
<i>in</i>	<i>section</i>	Which section we are going to look in
<i>in</i>	<i>key</i>	What the key name is

#### Returns

Returns the variable or L"" on failure.

#### Remarks

If the string is longer than `MAX_PATH*20` it will return an empty string.

#### 1.20.2.20 `DWORD writeProfileInfo ( bool user, std::wstring section, std::wstring key, std::wstring data )`

Writes a string in one of the applications program files.



1. Author Robin Stenvi

#### Parameters

in	<i>user</i>	Says whether we need to write the data to the user file or global file
in	<i>section</i>	Which section we are going to write in
in	<i>key</i>	What the key name is
in	<i>data</i>	What we are going to write

#### Returns

Returns 0 on success, otherwise -1 or GetLastError().

#### Remarks

Will now fail on system file, because we don't have privileges.

#### 1.20.2.21 std::string generateRandomString ( std::string *alphabet*, int *length* )

Generates a random string.

1. Author Robin Stenvi

#### Parameters

in	<i>alphabet</i>	Which character set for our random string
in	<i>length</i>	The length of the random string

#### Returns

Returns the random string that was generated

#### Remarks

srand() must have been called before this function.

#### 1.20.2.22 bool openFile ( GUID *dir*, std::wstring *dir2*, std::wstring *name*, CStdioFile & *file*, std::wstring & *outName*, UINT *permissions* )

Opens a file in a given Windows directory.

1. Author Robin Stenvi

#### Parameters

in	<i>dir</i>	The constant referring to a Windows path, using the function SHGetKnownFolderPath()
in	<i>dir2</i>	Any subdirectory that comes after dir, could be empty, should not have "\" at the end
in	<i>name</i>	The name of the file.
out	<i>file</i>	The CFile object that is opened.
out	<i>outName</i>	The full path to the file that has been opened.
in	<i>permissions</i>	The permissions for the file, default is CFile::modeCreate   CFile::modeWrite   CFile::modeRead.

**Returns**

Returns true if we succeed, false if we fail.

**1.20.2.23 std::string printUSASCII ( std::string & str )**

Make sure that the string is validated, in USASCII, according to the Syslog protocol.

This character set is used for hostname, app-name, process id, msg id, SD-ID. This does not handle size, it should be handled when they are created. If we see a character that doesn't fit we replace it with `replaceInvalid`.

1. Author Robin Stenvi

**Parameters**

<code>in, out</code>	<code>str</code>	The string that should abide by the following rule.
----------------------	------------------	---

**Returns**

Returns the new string

**1.20.2.24 std::string base64Encode ( std::vector< BYTE > inputBuffer )**

Converts a string to base 64, without padding.

The algorithm and code is gathered from [http://en.wikibooks.org/wiki/Algorithm\\_Implementation/Miscellaneous/Base64](http://en.wikibooks.org/wiki/Algorithm_Implementation/Miscellaneous/Base64). Has been modified slightly to suit our purposes, most notable is to remove the padding. We also changed the `'` character to `-`, so it's safe as a filename.

1. Author Robin Stenvi

**Parameters**

<code>in</code>	<code>inputBuffer</code>	The byte string that should be converted.
-----------------	--------------------------	---

**Returns**

Returns the base64 string.

## 2 Class Documentation

### 2.1 AboutDialog Class Reference

Sets the text and gets the version number from the registry.

#### Public Member Functions

- [AboutDialog](#) (CWnd \*pParent=NULL)  
*Constructor for the dialog.*
- virtual [~AboutDialog](#) ()  
*Destructor for the dialog.*
- virtual INT\_PTR [DoModal](#) ()  
*Makes the dialog modal.*

#### Protected Member Functions

- virtual void [DoDataExchange](#) (CDataExchange \*pDX)  
*Initates data exchange for elements.*

#### Private Attributes

- std::wstring [text](#)  
*string for holding the string temporary*
- CEdit [outText](#)  
*textfield for outputting text*

#### 2.1.1 Detailed Description

Sets the text and gets the version number from the registry.

This class creates, maintains and deletes the about BeLT dialog created when the user clicks the About BeLT button in the main window. It prints out a small text about the app and its creators before printing the applications version number retrieved from the registry.

1. Author Magnus Øverbø - 08.03.2013

#### 2.1.2 Constructor & Destructor Documentation

##### 2.1.2.1 AboutDialog::AboutDialog ( CWnd \* pParent = NULL )

Constructor for the dialog.

1. Author Magnus Øverbø

#### Parameters

in	<i>pParent</i>	Sent to parent.
----	----------------	-----------------

**2.1.2.2 AboutDialog::~~AboutDialog ( )** [virtual]

Destructor for the dialog.

1. Author Magnus Øverbø

**2.1.3 Member Function Documentation****2.1.3.1 void AboutDialog::DoDataExchange ( CDataExchange \* pDX )** [protected],[virtual]

Initates data exchange for elements.

1. Author Magnus Øverbø

**Parameters**

in	<i>pDX</i>	Sent to CDialogEx::DoDataExchange()
----	------------	-------------------------------------

**2.1.3.2 INT\_PTR AboutDialog::DoModal ( )** [virtual]

Makes the dialog modal.

1. Author Automatically generated

**Returns**

Returns CDialog::DoModal()

**2.2 Blacklist Struct Reference**

Whenever we see a value we are unable to retrieve we can use this to maybe save some time.

**2.2.1 Detailed Description**

Whenever we see a value we are unable to retrieve we can use this to maybe save some time.

Not used.

**2.3 Cbelt\_mainApp Class Reference**

Class that defines the application starting point, does not show a UI.

**Public Member Functions**

- [Cbelt\\_mainApp \( \)](#)  
*Constructor that is first called.*
- virtual BOOL [InitInstance \( \)](#)  
*Cbelt\_mainApp initialization.*

### Public Attributes

- HICON [m\\_plconList](#) [4]  
*Array containing the icons for the system tray.*

### 2.3.1 Detailed Description

Class that defines the application starting point, does not show a UI.

1. Author Automatically Generated

### 2.3.2 Constructor & Destructor Documentation

#### 2.3.2.1 Cbelt\_mainApp::Cbelt\_mainApp ( )

Constructor that is first called.

It first checks if there is an available update if there is, it will update automatically. Call the application with an argument that says "noupdate" to avoid automatic update.

1. Author Automatically Generated Robin Stenvi

### 2.3.3 Member Function Documentation

#### 2.3.3.1 BOOL Cbelt\_mainApp::InitInstance ( ) [virtual]

Cbelt\_mainApp initialization.

1. Author Automatically Generated Magnus Øverbø

### Returns

Always returns FALSE.

## 2.4 Cbelt\_mainDlg Class Reference

Main dialog window that is displayed to the user.

### Public Member Functions

- [Cbelt\\_mainDlg](#) (CWnd \*pParent=NULL)  
*Cbelt\_mainDlgs constructor.*
- [~Cbelt\\_mainDlg](#) ()  
*Destructor for Cbelt\_mainDlg.*
- [afx\\_msg void OnBnClickedButtonSendData](#) ()  
*Opens the transmission dialog for stored files.*

## Protected Member Functions

- virtual void [DoDataExchange](#) (CDataExchange \*pDX)  
*Creates data exchange for elements.*
- void [CANCEL](#) ()  
*Deletes the object in the tray and destroys the window.*
- void [OnGetMinMaxInfo](#) (MINMAXINFO FAR \*lpMMI)  
*Sets the minimum size of the main window.*
- virtual BOOL [OnInitDialog](#) ()  
*Initializes the dialog.*
- void [AppendLine](#) (CString str, COLORREF color)  
*Append a line to the screen using the color specified, caller need to supply newline.*
- void [updateStat](#) (BOOL X)  
*updateStat edits the mouse or keyboard counter*
- void [manageRawInput](#) (HWND me)  
*Register for raw input messages, so we can detect different keyboards or mouses.*
- void [sendKeyboardInfo](#) ()  
*Retrives information about the keyboard and send it to the server.*
- afx\_msg LRESULT [printEvent](#) (WPARAM w, LPARAM l)  
*Prints the string specified if the user has specified it.*
- afx\_msg LRESULT [changePause](#) (WPARAM w, LPARAM l)  
*Changes the pause state of the app.*
- afx\_msg void [OnSysCommand](#) (UINT nID, LPARAM lParam)  
*Runs the system commands.*
- afx\_msg void [OnPaint](#) ()  
*If you add a minimize button to your dialog, you will need the code below to draw the icon.*
- afx\_msg HCURSOR [OnQueryDragIcon](#) ()  
*The system calls this function to obtain the cursor to display while the user drags the minimized window.*
- afx\_msg LRESULT [OnTrayNotify](#) (WPARAM wParam, LPARAM lParam)  
*Perform actions based on interaction with the system tray icon.*
- afx\_msg void [OnTrayRestore](#) ()  
*If hidden restore the window and set it to be shown.*
- afx\_msg void [OnHideapp](#) ()  
*This function runs the OnHide, and then hides the application.*
- afx\_msg void [OnDestroy](#) ()  
*Destroys the window by deleting the object and setting it to NULL.*
- afx\_msg void [OnAppExit](#) ()  
*Upon exit from the tray icon, stop the listener and quit the application, which destroys the window.*
- afx\_msg void [OnSize](#) (UINT nType, int cx, int cy)  
*Resizing the window along with the RichEditControl field.*
- afx\_msg LRESULT [OnQueryEndSession](#) (WPARAM wParam, LPARAM lParam)  
*Responds to the OnQuerySessionEnd.*
- afx\_msg LRESULT [OnEndSession](#) (WPARAM wParam, LPARAM lParam)  
*Exits the application in response to the OnEndSession message.*
- afx\_msg LRESULT [OnPowerState](#) (WPARAM wParam, LPARAM lParam)  
*Handles when the system is suspended.*
- afx\_msg LRESULT [OnClose](#) (WPARAM w, LPARAM l)  
*Hide the application when clicking the X in the top right corner.*
- afx\_msg void [OnSetDlg](#) ()  
*Opens the settings dialog for BeLT.*
- afx\_msg BOOL [OnDeviceChange](#) (UINT nEventType, DWORD\_PTR dwData)

See <http://msdn.microsoft.com/en-us/library/windows/desktop/aa363205%28v=vs.85%29.aspx> for more events.

- `afx_msg LRESULT FAIL` (WPARAM w, LPARAM l)
 

*When something fails elsewhere in the application, this will notify the users, prints the string in red.*
- `afx_msg LRESULT SUCCESS` (WPARAM w, LPARAM l)
 

*Need to notify the user of something that has succeeded, will print the string in green.*
- `afx_msg void OnAppAbout` ()
 

*Initiates the about dialog described in the resource file.*
- `afx_msg LRESULT setIcon` (WPARAM w, LPARAM l)
 

*Sets the icon if a password field is detected.*
- `afx_msg void OnRawInput` (UINT nInputcode, HRAWINPUT hRawInput)
 

*Called each time we receive raw input, which we have registered for.*
- `afx_msg void controlListener` ()
 

*Changes the start/pause/resume buttons functionality.*
- `afx_msg void stopListener` ()
 

*Stops all listeners, if they are listening, otherwise it does nothing.*
- `afx_msg void sendToServer` ()
 

*When something fails elsewhere in the application, this will notify the user.*
- `afx_msg void setFilterSettings` ()
 

*Opens the filter settings dialog.*
- `void startListener` ()
 

*Starts listener, if it's not listening or is paused.*
- `void pauseListener` ()
 

*Pauses the listener, if we are listening, otherwise it does nothing.*
- `void resumeListener` ()
 

*Resumes listening after pause, if it was not paused it does nothing.*

#### Static Protected Member Functions

- static DWORD WINAPI `monitorHWUsage` (void \*s)
 

*Function for monitoring the hardware usage for CPU and RAM.*
- static UINT `startKeylog` (LPVOID param)
 

*Starts mouse and keyboard hooking, returns NULL, and unhook if one failed, 1 if we succeed.*

#### Private Attributes

- bool `listening`

*If we running the listener or not.*
- int `mouseCount`

*Counter for mouse click and keystrokes.*
- `AboutDialog * aboutDlg`

*Dialog that show about message.*
- `filterSettings * filters`

*Dialog that lets the user choose which events he want to see.*
- UINT `settings`

*Current setting for which events to display.*
- `HIDDevice hidDevices [MAX_HIDS]`

*An entry for each discovered device.*
- int `currHids`

*Number of devices we have seen so far.*

- HANDLE [lastKey](#)  
*The last mouse and jey we saw, used to discover changes.*
- CTrayNot \* [m\\_pTray](#)  
*pointer for a object in the system tray*
- BOOL [m\\_bHidden](#)  
*States wether the window is hidden.*
- HANDLE [HWMonThread](#)  
*Handles to deal with hardware monitoring thread.*
- DWORD [HWMonThreadId](#)  
*Thread ID to deal with hardware monitoring thread.*
- HANDLE [keyThreadHandle](#)  
*Handle to deal with keyboard thread.*
- HANDLE [mouseThreadHandle](#)  
*Handle to deal with mouse thread.*
- HICON [m\\_hIcon](#)  
*Icon for the application.*
- CRichEditCtrl [richEdit](#)  
*Declares the richEditControl.*
- CEdit \* [countKey](#)  
*Declares the statistics fields.*
- CButton \* [ctrlButton](#)  
*Declares pointer for a button for start/pause/resume.*
- CButton [serverCheckBox](#)  
*Button if we want to send to server.*

#### Static Private Attributes

- static const int [MAX\\_HIDS](#) = 100  
*Max number of possible devices.*

#### 2.4.1 Detailed Description

Main dialog window that is displayed to the user.

This class handles almost everything that is displayed to the user, and initiates almost everything:

- Displaying events in a text field
- Controlling start/stop and pause/resume via buttons
- Starting and stopping connection with the server
- Initiate filter settings dialog
- Initiate about dialog
- Handle RAW input for detecting mouse HW changes
- Handle the interval for when we check HW usage
- Send information about the keyboard
- Handling the tray icon
- Recording user statistict regarding input events



1. Author Automatically generated  
Magnus Øverbø

Robin Stenvi

## 2.4.2 Constructor & Destructor Documentation

### 2.4.2.1 Cbelt\_mainDlg::Cbelt\_mainDlg ( CWnd \* *pParent* = NULL )

Cbelt\_mainDlg constructor.

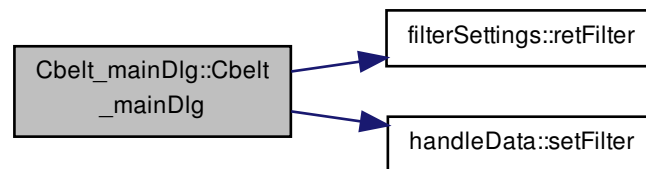
1. Author Automatically Generated  
Robin Stenvi

Magnus Øverbø

#### Parameters

<i>in</i>	<i>pParent</i>	Sent to parent.
-----------	----------------	-----------------

Here is the call graph for this function:



### 2.4.2.2 Cbelt\_mainDlg::~~Cbelt\_mainDlg ( )

Destructor for Cbelt\_mainDlg.

1. Author Automatically Generated  
Robin Stenvi

Magnus Øverbø

## 2.4.3 Member Function Documentation

### 2.4.3.1 void Cbelt\_mainDlg::DoDataExchange ( CDataExchange \* *pDX* ) [protected], [virtual]

Creates data exchange for elements.

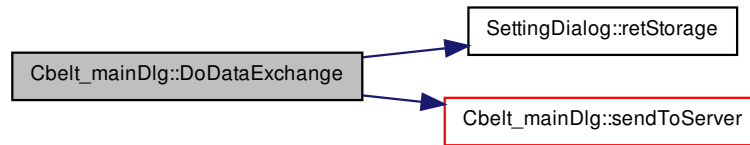
1. Author Automatically Generated

Robin Stenvi

#### Parameters

<i>in</i>	<i>pDX</i>	Sent to CDialogEx::DoDataExchange().
-----------	------------	--------------------------------------

Here is the call graph for this function:



#### 2.4.3.2 void Cbelt\_mainDlg::CANCEL ( ) [protected]

Deletes the object in the tray and destroys the window.

1. Author Magnus Øverbø - 14.02.2013

Here is the call graph for this function:



#### 2.4.3.3 DWORD WINAPI Cbelt\_mainDlg::monitorHWUsage ( void \* s ) [static], [protected]

Function for monitoring the hardware usage for CPU and RAM.

1. Author Robin Stenvi

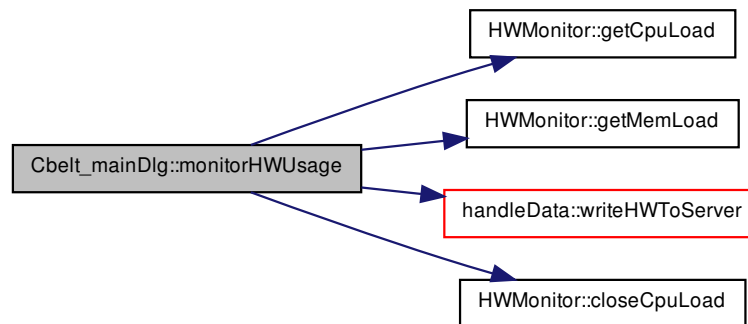
##### Parameters

in	s	Not used.
----	---	-----------

**Returns**

Always returns 0.

Here is the call graph for this function:



#### 2.4.3.4 `UINT Cbelt_mainDlg::startKeylog ( LPVOID param )` `[static]`, `[protected]`

Starts mouse and keyboard hooking, returns NULL, and unhook if one failed, 1 if we succeed.

1. Author Robin Stenvi

**Parameters**

<code>in</code>	<code>param</code>	Unused.
-----------------	--------------------	---------

**Returns**

Returns 0 if we faile, 1 if we succeed.

#### 2.4.3.5 `void Cbelt_mainDlg::OnGetMinMaxInfo ( MINMAXINFO FAR * lpMMI )` `[protected]`

Sets the minimum size of the main window.

1. Author Robin Stenvi

#### 2.4.3.6 `BOOL Cbelt_mainDlg::OnInitDialog ( )` `[protected]`, `[virtual]`

Initializes the dialog.

1. Author Automatically Generated  
Robin Stenvi

Magnus Øverbø

**Returns**

Always returns TRUE.

Here is the call graph for this function:



#### 2.4.3.7 void Cbelt\_mainDlg::AppendLine ( CString *str*, COLORREF *color* ) [protected]

Append a line to the screen using the color specified, caller need to supply newline.

1. Author Robin Stenvi

**Parameters**

in	<i>str</i>	The string that should be printed, should end in
in	<i>color</i>	Specified with RGB(x,y,z)

#### 2.4.3.8 void Cbelt\_mainDlg::updateStat ( BOOL *X* ) [protected]

updateStat edits the mouse or keyboard counter

1. Author Magnus Øverbø

**Parameters**

in	<i>X</i>	States which of the fields to update
----	----------	--------------------------------------

#### 2.4.3.9 void Cbelt\_mainDlg::manageRawInput ( HWND *me* ) [protected]

Register for raw input messages, so we can detect different keyboards or mouses.

1. Author Robin Stenvi

**Parameters**

in	<i>me</i>	HWND to the main dialog window.
----	-----------	---------------------------------

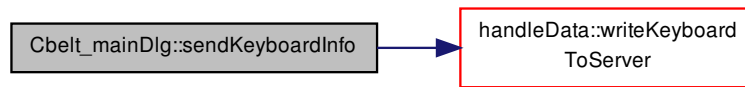
#### 2.4.3.10 void Cbelt\_mainDlg::sendKeyboardInfo ( ) [protected]

Retrives information about the keyboard and send it to the server.

[http://msdn.microsoft.com/en-us/library/windows/desktop/ms724336\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms724336(v=vs.85).aspx) [http://msdn.microsoft.com/en-us/library/windows/desktop/dd318691\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/dd318691(v=vs.85).aspx)

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.4.3.11 afx\_msg LRESULT Cbelt\_mainDlg::printEvent ( WPARAM w, LPARAM l ) [protected]

Prints the string specified if the user has specified it.

1. Author Robin Stenvi

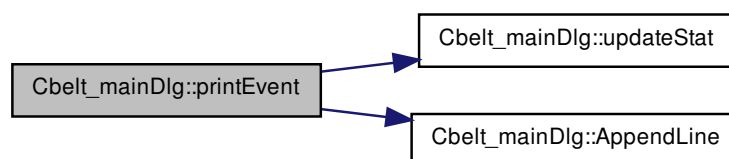
##### Parameters

in	w	CString pointer allocated with new and is the string that should be printed, should end in newline. This function deletes this value.
in	l	UINT value specifying what type of event it is.

##### Returns

Always returns 0.

Here is the call graph for this function:



#### 2.4.3.12 afx\_msg LRESULT Cbelt\_mainDlg::changePause ( WPARAM w, LPARAM l ) [protected]

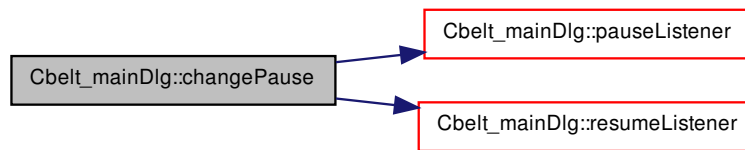
Changes the pause state of the app.

1. Author Robin Stenvi

##### Parameters

in	w	BOOL value that is FALSE if we should pause, TRUE if we should resume.
in	l	Unused.

Here is the call graph for this function:



#### 2.4.3.13 void Cbelt\_mainDlg::OnSysCommand ( UINT *nID*, LPARAM *lParam* ) [protected]

Runs the system commands.

1. Author Automatically generated

##### Parameters

in	<i>nID</i>	First argument to <code>CDialogEx::OnSysCommand()</code> .
in	<i>lParam</i>	Second argument to <code>CDialogEx::OnSysCommand()</code> .

#### 2.4.3.14 void Cbelt\_mainDlg::OnPaint ( ) [protected]

If you add a minimize button to your dialog, you will need the code below to draw the icon.

For MFC applications using the document/view model, this is automatically done for you by the framework

1. Author Automatically generated Magnus Øverbø

Here is the call graph for this function:



#### 2.4.3.15 HCURSOR Cbelt\_mainDlg::OnQueryDragIcon ( ) [protected]

The system calls this function to obtain the cursor to display while the user drags the minimized window.

1. Author Robin Stenvi

##### Returns

Returns `static_cast<HCURSOR>(m_hIcon)`.

#### 2.4.3.16 LRESULT Cbelt\_mainDlg::OnTrayNotify ( WPARAM *wParam*, LPARAM *lParam* ) [protected]

Perform actions based on interaction with the system tray icon.

When interacting with the tray icon create a pop-up menu on right click and put the window to the foreground and restore it. On double click set the window to the foreground or restore it.

1. Author Magnus Øverbø - 14.02.2013

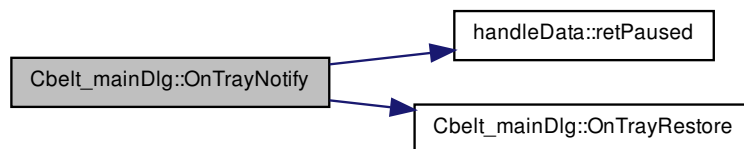
##### Parameters

in	<i>wParam</i>	wParam is not used
in	<i>lParam</i>	lParam contains what type of click was performed

##### Returns

Always returns 0.

Here is the call graph for this function:



#### 2.4.3.17 void Cbelt\_mainDlg::OnTrayRestore ( ) [protected]

If hidden restore the window and set it to be shown.

1. Author Magnus Øverbø - 14.02.2013

#### 2.4.3.18 void Cbelt\_mainDlg::OnHideapp ( ) [protected]

This function runs the OnHide, and then hides the application.

1. Author Magnus Øverbø - 04.02.2013

Here is the call graph for this function:



**2.4.3.19 void Cbelt\_mainDlg::OnDestroy ( )** [protected]

Destroys the window by deleting the object and setting it to NULL.

1. Author Magnus Øverbø - 14.02.2013

**2.4.3.20 void Cbelt\_mainDlg::OnAppExit ( )** [protected]

Upon exit from the tray icon, stop the listener and quit the application, which destroys the window.

1. Author Magnus Øverbø - 14.02.2013

Here is the call graph for this function:

**2.4.3.21 void Cbelt\_mainDlg::OnSize ( UINT nType, int cx, int cy )** [protected]

Resizing the window along with the RichEditControl field.

Borrowed from the Ftpree MSDN sample. When clicking the minimize button the application is automatically hidden instead.

1. Author Magnus Øverbø - 14.02.2013

Magnus Øverbø - 25.03.2013

**Parameters**

in	<i>nType</i>	which states what type of event has occurred
in	<i>cx</i>	int which states the width of the window
in	<i>cy</i>	int which states the height of the window

Here is the call graph for this function:

**2.4.3.22 LRESULT Cbelt\_mainDlg::OnQueryEndSession ( WPARAM wParam, LPARAM lParam )** [protected]

Responds to the OnQuerySessionEnd.



1. Author Magnus Øverbø - 14.02.2013

#### Parameters

in	<i>wParam</i>	wParam is not used
in	<i>lParam</i>	lParam contains the type of event that is occurring

#### Returns

true as defined by the windows api

#### 2.4.3.23 HRESULT Cbelt\_mainDlg::OnEndSession ( WPARAM *wParam*, LPARAM *lParam* ) [protected]

Exits the application in response to the OnEndSession message.

1. Author Magnus Øverbø - 27.02.2013

#### Parameters

in	<i>wParam</i>	wParam is true when the challenge received true
in	<i>lParam</i>	lParam contains information about the event that occur

#### Returns

0 as per definition by the windows API

Here is the call graph for this function:



#### 2.4.3.24 HRESULT Cbelt\_mainDlg::OnPowerState ( WPARAM *wParam*, LPARAM *lParam* ) [protected]

Handles when the system is suspended.

It receives the WM\_POWERBROADCAST message which is relayed to this function that based on the WPARAM parameter decides what to do. It only responds to the PBT\_APMRESUMEAUTOMATIC, which is sent when coming back from a suspension, and PBT\_APMSUSPEND, which is sent when going into a suspended state.

1. Author Magnus Øverbø - 05.04.2013

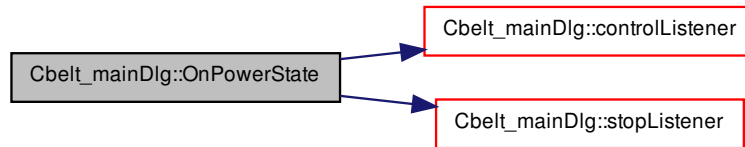
#### Parameters

in	<i>wParam</i>	WPARAM with the ID of a specific event
in	<i>lParam</i>	LPARAM with message specificic to the WPARAM event

**Returns**

TRUE BOOL value that has to be the response

Here is the call graph for this function:



### 2.4.3.25 LRESULT Cbelt\_mainDlg::OnClose ( WPARAM w, LPARAM l ) [protected]

Hide the application when clicking the X in the top right corner.

if we have used the exit function in a menu we close the application using the `OnAppExit` function. if we have received a message generated by clicking the exit function in the top right corner(exit button), we redirect the function and call the `OnHideapp` function to hide the application instead. Usually the `WM_CLOSE` contains nothing in its parameters, so we've used the values 3 and 1 just to have a little buffer zone if microsoft decides to use it.

1. Author Magnus Øverbø - 19.04.2013

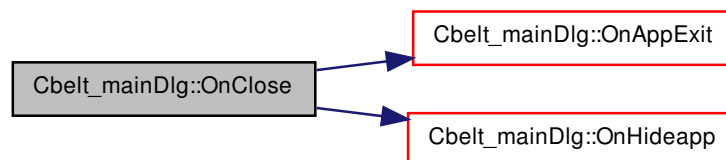
**Parameters**

in	w	is 3 if we are closing the application, otherwise its not used
in	l	is 1 if we are closing the application, otherwise its not used

**Returns**

LRESULT is returned as a 0 as according tho the standard by mdsn

Here is the call graph for this function:

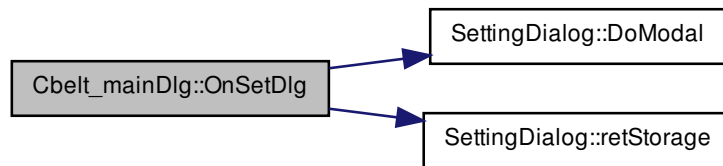


### 2.4.3.26 void Cbelt\_mainDlg::OnSetDlg ( ) [protected]

Opens the settings dialog for BeLT.

1. Author Magnus Øverbø - 25.03.2013

Here is the call graph for this function:



**2.4.3.27** `afx_msg BOOL Cbelt_mainDlg::OnDeviceChange ( UINT nEventType, DWORD_PTR dwData )`  
 [protected]

See <http://msdn.microsoft.com/en-us/library/windows/desktop/aa363205%28v=vs.85%29.aspx> for more events.

1. Author Robin Stenvi

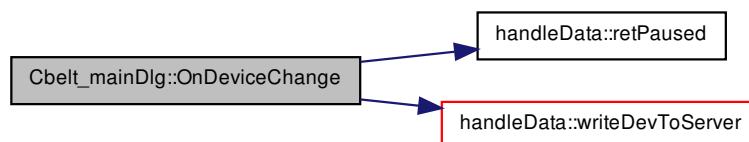
#### Parameters

in	<i>nEventType</i>	What type of event has happened (supplied by Windows)
in	<i>dwData</i>	Information about the device.

#### Returns

Returns `CWnd::OnDeviceChange (nEventType, dwData)`.

Here is the call graph for this function:



**2.4.3.28** `afx_msg LRESULT Cbelt_mainDlg::FAIL ( WPARAM w, LPARAM l )` [protected]

When something fails elsewhere in the application, this will notify the users, prints the string in red.

1. Author Robin Stenvi

#### Parameters

in	w	CString pointer allocated with new, this is the line that should be printed to the user. this function will delete the pointer
in	/	UINT that can be used to mark additional actions that should be taken, if no action should be taken, use 0

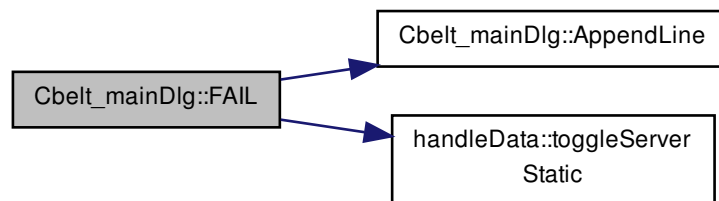
#### Returns

Always returns 0.

#### Remarks

If SSL connection fails, we keep trying, until we get it. The function in handleData will handle delay.

Here is the call graph for this function:



#### 2.4.3.29 afx\_msg LRESULT Cbelt\_mainDlg::SUCCESS ( WPARAM w, LPARAM l ) [protected]

Need to notify the user of something that has succeeded, will print the string in green.

1. Author Robin Stenvi

Magnus Øverbø

#### Parameters

in	w	CString pointer allocated with new, this is the line that should be printed to the user. this function will delete the pointer
in	/	UINT that can be used to mark additional actions that should be taken, if no action should be taken, use 0

**Returns**

Always returns 0.

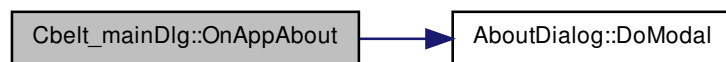
Here is the call graph for this function:

**2.4.3.30 void Cbelt\_mainDlg::OnAppAbout ( ) [protected]**

Initiates the about dialog described in the resource file.

1. Author Magnus Øverbø - 27.02.2013

Here is the call graph for this function:

**2.4.3.31 afx\_msg LRESULT Cbelt\_mainDlg::setIcon ( WPARAM w, LPARAM l ) [protected]**

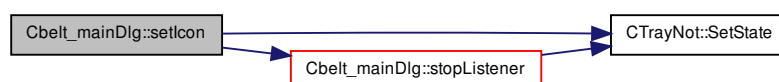
Sets the icon if a password field is detected.

1. Author Robin Stenvi Magnus Øverbø

**Parameters**

in	w	BOOL value, where TRUE means that we have paused
in	l	Unused

Here is the call graph for this function:



### 2.4.3.32 void Cbelt\_mainDlg::OnRawInput ( UINT *nInputcode*, HRAWINPUT *hRawInput* ) [protected]

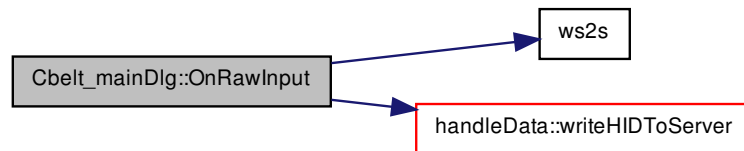
Called each time we receive raw input, which we have registered for.

1. Author Robin Stenvi

#### Parameters

in	<i>nInputcode</i>	Determines if the input happened while the application was in the foreground or not. RIM_INPUT means it was in the foreground, RIM_INPUTSINK means it was not in the foreground
in	<i>hRawInput</i>	Handle to the RAWINPUT structure

Here is the call graph for this function:



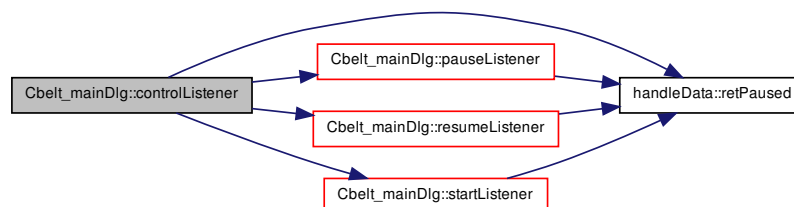
### 2.4.3.33 void Cbelt\_mainDlg::controlListener ( ) [protected]

Changes the start/pause/resume buttons functionality.

At first it is set to "start" then upon starting the loggin it is set to "Pause". After this interaction it is set to "resume", which again upon interaction is set to "pause". When interacted with it calls its corresponding function dependant on BeLTs state.

1. Author Magnus Øverbø - 11.03.2013

Here is the call graph for this function:



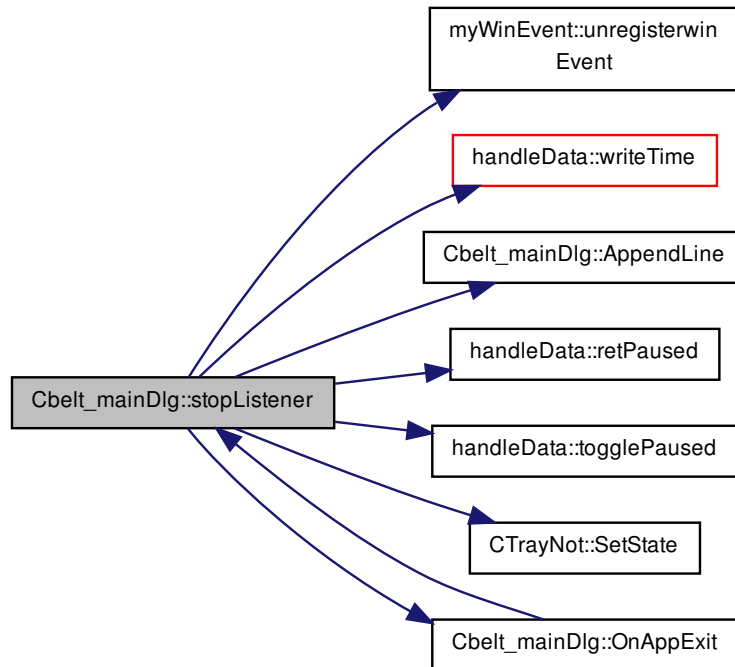
### 2.4.3.34 void Cbelt\_mainDlg::stopListener ( ) [protected]

Stops all listeners, if they are listening, otherwise it does nothing.

1. Author Robin Stenvi

Magnus Øverbø

Here is the call graph for this function:

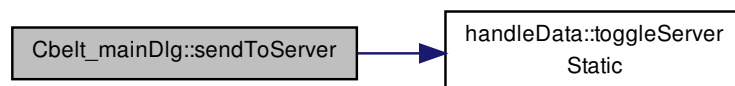


#### 2.4.3.35 void Cbelt\_mainDlg::sendToServer ( ) [protected]

When something fails elsewhere in the application, this will notify the user.

1. Author Robin Stenvi

Here is the call graph for this function:

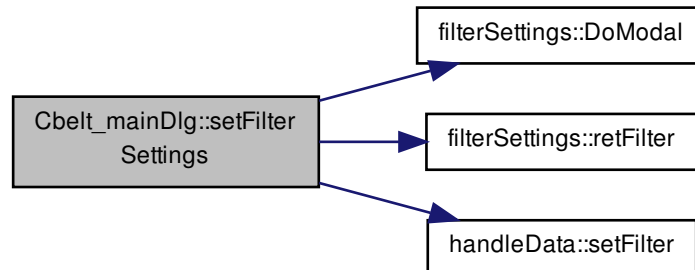


#### 2.4.3.36 void Cbelt\_mainDlg::setFilterSettings ( ) [protected]

Opens the filter settings dialog.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.4.3.37 void Cbelt\_mainDlg::startListener ( ) [protected]

Starts listener, if it's not listening or is paused.

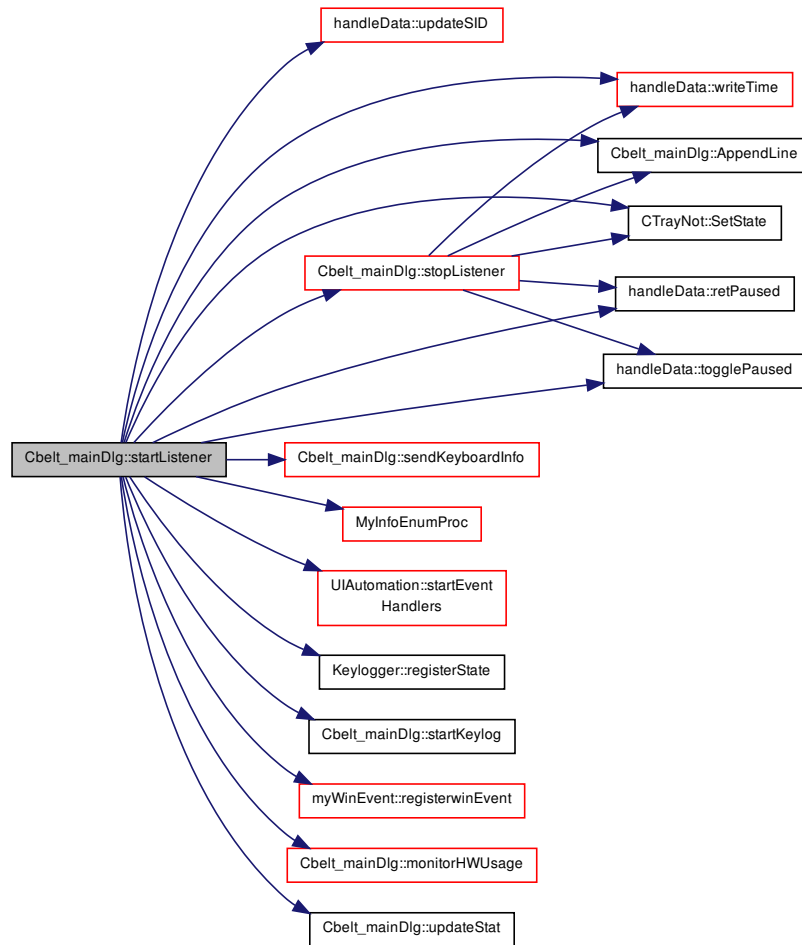
1. Author Robin Stenvi

#### Remarks

If anything fails, we just make sure the error doesn't propagate, but we still continue logging with whatever we have left. The only exception is if `writeTime` in `handleData` fails, then we only count up the session number.



Here is the call graph for this function:

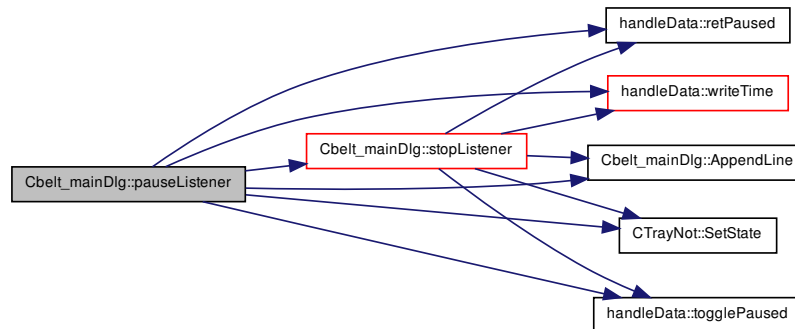


#### 2.4.3.38 void Cbelt\_mainDlg::pauseListener ( ) [protected]

Pauses the listener, if we are listening, otherwise it does nothing.

1. Author Robin Stenvi

Here is the call graph for this function:



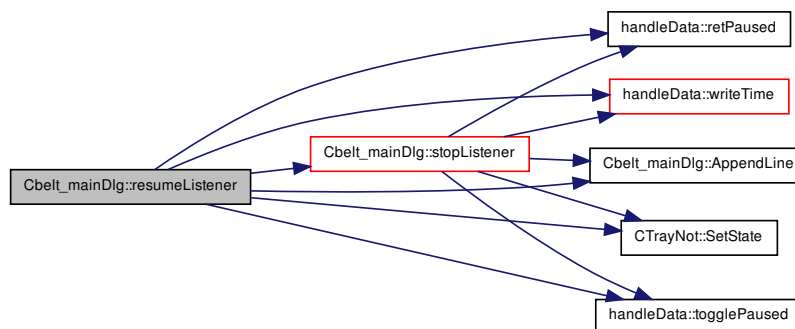
#### 2.4.3.39 void Cbelt\_mainDlg::resumeListener ( ) [protected]

Resumes listening after pause, if it was not paused it does nothing.

1. Author Robin Stenvi

Magnus Øverbø

Here is the call graph for this function:



#### 2.4.3.40 void Cbelt\_mainDlg::OnBnClickedButtonSendData ( )

Opens the transmission dialog for stored files.

1. Author Robin Stenvi

## 2.5 checkUpdate Class Reference

Class that checks if a new update is available.

### Public Member Functions

- [checkUpdate](#) ( )

- Empty constructor.*
- [ERRORS check](#) ()  
*Checks and install new update.*

### Protected Member Functions

- [ERRORS initTLS](#) ()  
*Initializes the TLS, call this in the beginning.*
- [ERRORS startTLS](#) ()  
*Starts all the necessary handlers for TLS connection.*
- void [closeTLS](#) ()  
*The opposite of initTLS()*
- int [newerVersion](#) (std::string current, std::string newVersion)  
*Returns 1 if the server version is newer.*
- std::vector< std::string > [explode](#) (const std::string &str, const char &ch)  
*Takes in an std::string and returns a vector where "ch" is the delimiter.*
- [ERRORS getVersionNum](#) (std::string &version)  
*Gets the version number from the registry.*
- [ERRORS getserverSettings](#) (std::vector< std::string > &settings)  
*Gets all the necessary data from the server config file and returns it in a vector of std::string.*
- [ERRORS initiateConnection](#) (std::string ip, int port)  
*Initiate the necessary sockets for use against the server.*
- [ERRORS getFile](#) (std::string ip, int port, std::string file, std::string &response)  
*Parses the file from the server and returns a vector with std::string containing all the necessary data.*
- int [closeConnection](#) ()  
*Closes an Internet connection.*
- [ERRORS writeFile](#) (std::string data, std::string fileName, std::wstring &filePath)  
*Writes contents to temporary folder.*
- bool [internetAvailable](#) ()  
*Checks if Internet is available.*

### Private Attributes

- WSADATA [wsaData](#)  
*Windows sockets data.*
- SOCKET [sock](#)  
*The socket we against the server.*
- hostent \* [h](#)  
*Information about the server.*
- sockaddr\_in [serveraddr](#)  
*Information about the server.*
- SSL \* [ssl](#)  
*The SSL object we need to tunnel our traffic over SSL.*
- SSL\_CTX \* [ctx](#)  
*Framework object to set different options.*
- bool [useTLS](#)  
*Whether we use TLS or not.*

### 2.5.1 Detailed Description

Class that checks if a new update is available.

This is a test.

Only one function is available from the outside, and that is the actual check for the update "check()". The remaining functions are just there to help the process.

1. Author Robin Stenvi

### 2.5.2 Constructor & Destructor Documentation

#### 2.5.2.1 checkUpdate::checkUpdate ( )

Empty constructor.

1. Author Robin Stenvi

### 2.5.3 Member Function Documentation

#### 2.5.3.1 **ERRORS** checkUpdate::initTLS ( ) [protected]

Initializes the TLS, call this in the beginning.

1. Author Robin Stenvi

#### Returns

Always returns OK.

#### 2.5.3.2 **ERRORS** checkUpdate::startTLS ( ) [protected]

Starts all the necessary handlers for TLS connection.

1. Author Robin Stenvi

#### Returns

Return OK if everythin went well, otherwise it returns an error value.

#### 2.5.3.3 void checkUpdate::closeTLS ( ) [protected]

The opposite of initTLS()

1. Author Robin Stenvi

#### 2.5.3.4 int checkUpdate::newerVersion ( std::string *current*, std::string *newVersion* ) [protected]

Returns 1 if the server version is newer.

Both the parameters need to be of the format X.Y.Z where X, Y, and Z is any integer. There is some undefined behaviour in atoi, should use a different function.

1. Author Robin Stenvi

#### Parameters

in	<i>current</i>	The current version number for the application
in	<i>newVersion</i>	What the server says is the newest version

#### Returns

Returns -1 (the current is a newer version), 0 (the same version) or 1 (newer version available)

**2.5.3.5** `std::vector< std::string > checkUpdate::explode ( const std::string & str, const char & ch )` [protected]

Takes in an std::string and returns a vector where "ch" is the delimiter.

1. Author Robin Stenvi

#### Parameters

in	<i>str</i>	The string we want to split.
in	<i>ch</i>	The delimiter

#### Returns

A vector that contains all the values.

**2.5.3.6** **ERRORS** `checkUpdate::getVersionNum ( std::string & version )` [protected]

Gets the version number from the registry.

1. Author Robin Stenvi

#### Parameters

out	<i>version</i>	Output value which should contain the version number when the function complete.
-----	----------------	--

#### Returns

An error value indicating how it went, can be OPEN\_REG, QUERY\_REG\_VALUE or OK.

**2.5.3.7** **ERRORS** `checkUpdate::getserverSettings ( std::vector< std::string > & settings )` [protected]

Gets all the necessary data from the server config file and returns it in a vector of std::string.

1. Author Robin Stenvi

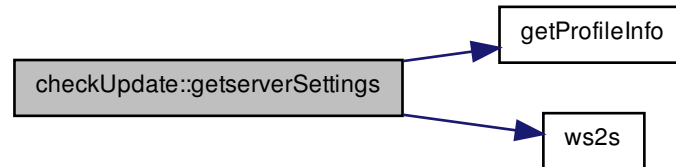
#### Parameters

out	<i>settings</i>	Output value containing all the values
-----	-----------------	--

**Returns**

Returns GET\_SETTINGS or OK.

Here is the call graph for this function:



### 2.5.3.8 ERRORS `checkUpdate::initiateConnection ( std::string ip, int port )` [protected]

Initiate the necessary sockets for use against the server.

1. Author Robin Stenvi

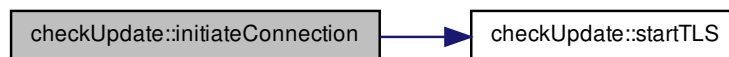
**Parameters**

<i>in</i>	<i>ip</i>	IP-address or domain name to the server.
<i>in</i>	<i>port</i>	Port number to the server.

**Returns**

Returns INIT\_SOCKET, INIT\_WSA or OK.

Here is the call graph for this function:



### 2.5.3.9 ERRORS `checkUpdate::getFile ( std::string ip, int port, std::string file, std::string & response )` [protected]

Parses the file from the server and returns a vector with `std::string` containing all the necessary data.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>response</i>	The response from the server, after HTTP headers have been removed.
-----------	-----------------	---

**Returns**

A vector of `std::string` containing all the relevant data that was found. Returns a file from the server over HTTP

1. Author Robin Stenvi

**Parameters**

in	<i>ip</i>	The IP-adress or domain to the server.
in	<i>port</i>	The port number on the server.
in	<i>file</i>	where on the server the file is (without domain name, but with leading /)
out	<i>response</i>	The file we get from the server, is returned to the caller.

**Returns**

The function returns a self-defined error message, which can be `CONNECT`, `SEND_DATA` or `OK`.

Here is the call graph for this function:



### 2.5.3.10 `int checkUpdate::closeConnection ( )` [protected]

Closes an Internet connection.

1. Author Robin Stenvi

**Returns**

Returns the value of `GetLastError()` if something goes wrong, otherwise it returns 0

### 2.5.3.11 `ERRORS checkUpdate::writeFile ( std::string data, std::string fileName, std::wstring & filePath )` [protected]

Writes contents to temporary folder.

1. Author Robin Stenvi

**Parameters**

in	<i>data</i>	The data that should be written, can be binary or ASCII
in	<i>fileName</i>	The filename for the file, can have trailing directories, this will be excluded
out	<i>filePath</i>	The final path to the file.

**Returns**

An error if something went wrong, or it returns OK.

**2.5.3.12** `bool checkUpdate::internetAvailable ( ) [protected]`

Checks if Internet is available.

1. Author Robin Stenvi

**Returns**

Returns true if Internet is available, false if Internet is unavailable or we are unable to check.

**2.5.3.13** **ERRORS** `checkUpdate::check ( )`

Checks and install new update.

First it retrieves the current version from the registry. Then it retrieves the server configuration. Then it checks whether it has a port number for SSL, if not it falls back on regular HTTP. Then it gets the newest server configuration from the server. Then it checks whether we have the newest version or not. If we don't have the newest version it will run `belt_update.exe` to get the newest version.

1. Author Robin Stenvi

**Returns**

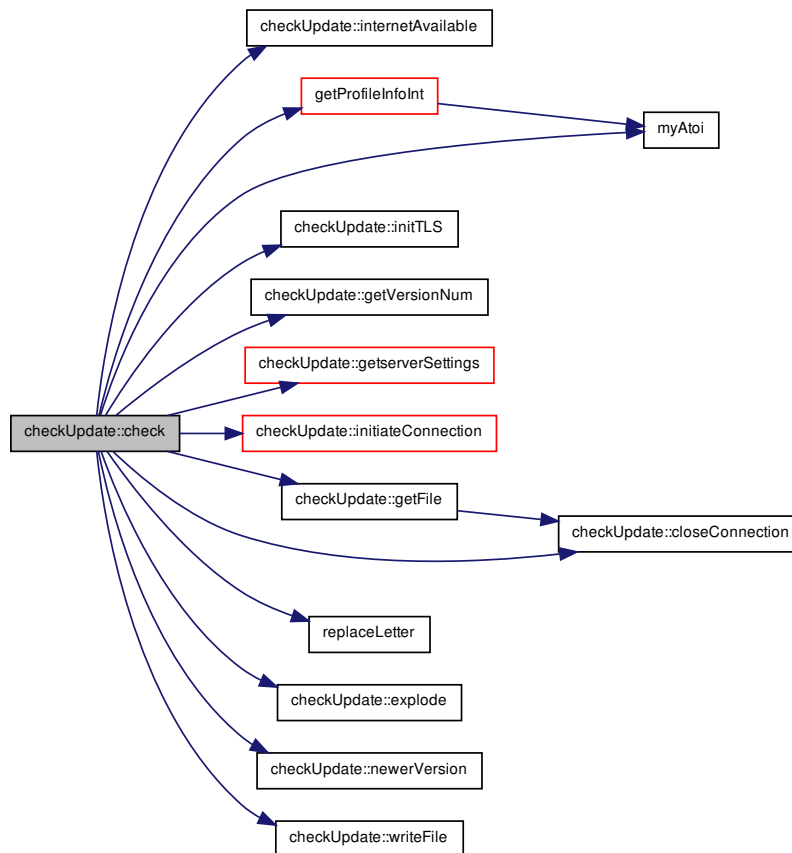
Returns an error message, OK or it exits the application for update.

**Remarks**

{Should change the offset thing before release, if there is no TLS port it should just be zero.}



Here is the call graph for this function:



## 2.6 CTrayNot Class Reference

A class for creating/maintaining the system tray icon.

### Public Member Functions

- [CTrayNot](#) (CWnd \*pWnd, UINT uCallbackMessage, LPCTSTR szTip, HICON \*pList)  
*Contstructor for the class, creates the initial system tray icon.*
- virtual [~CTrayNot](#) ()  
*Destroys the CTrayNot object by sending a NIM\_DELETE to the Shell\_NotifyIcon function.*
- bool [SetState](#) (int id=0)  
*SetState changes the icon in the system tray.*

### Private Attributes

- BOOL [m\\_bEnabled](#)  
*Declares if the icon is enabled.*
- NOTIFYICONDATA [m\\_tnd](#)  
*The structure for the system tray icon.*

- HICON \* [m\\_plconList](#)  
*List of all the icons available.*

### 2.6.1 Detailed Description

A class for creating/maintaining the system tray icon.

This class creates the object which are displayed in the system tray. The class has three variables and three functions for creating, destroying and changing the state of the object. The class is a modified version of the example project at <http://www.codeproject.com/Articles/1627/System-Tray--Icons-Adding-to-your-dialog-applicatio> made by Ash Rowe. This class creates a system tray notification object, by passing a notification data structure to Shell\_NotifyIcon. Then, later on it destroys it by sending a NIM\_DELETE to the Shell\_NotifyIcon. To update it the SetState function changes the current icon and sends a NIM\_MODIFY to the Shell\_NotifyIcon.

1. Author Magnus Øverbø

#### Date

2013-01-16 - 2012-01-31 (Last modified)

### 2.6.2 Constructor & Destructor Documentation

#### 2.6.2.1 CTrayNot::CTrayNot ( CWnd \* pWnd, UINT uCallbackMessage, LPCTSTR szTip, HICON \* pList )

Constructor for the class, creates the initial system tray icon.

This function creates a system tray notification object, by passing a notification data structure to Shell\_NotifyIcon.

1. Author Magnus Øverbø - 31.01.2013

#### 2.6.2.2 CTrayNot::~CTrayNot ( ) [virtual]

Destroys the CTrayNot object by sending a NIM\_DELETE to the Shell\_NotifyIcon function.

1. Author Magnus Øverbø, 31.01.2013

### 2.6.3 Member Function Documentation

#### 2.6.3.1 bool CTrayNot::SetState ( int id = 0 )

SetState changes the icon in the system tray.

The id argument states which icon to be shown.

1. Author Magnus Øverbø, 31.01.2013 Magnus Øverbø, 09.05.2013

#### Parameters

in	<i>id</i>	The icon that should be set, RED_ICON, GREEN_ICON, BLUE_ICON or YELLOW_ICON.
----	-----------	--

**Returns**

returns true if the icon was changed and false if it failed to change the icon after ten tries

**2.7 deviceInfo Struct Reference**

Information about a storage device, only a timestamp and a value that says whether it was inserted or removed.

**Public Attributes**

- `std::string` [desc](#)  
*Says whether it was removed or inserted.*
- `DWORD` [time](#)  
*The moment we registered it.*

**2.7.1 Detailed Description**

Information about a storage device, only a timestamp and a value that says whether it was inserted or removed.

**2.8 eventHandler Class Reference**

Responsible for registering for events and receiving these events.

**Public Member Functions**

- [eventHandler](#) ()  
*Initializes all variables to NULL and says that eventhandler has not been added yet.*
- [~eventHandler](#) ()  
*Empty destructor.*
- `ULONG STDMETHODCALLTYPE` [AddRef](#) ()  
*One of the IUnknown methods we need to implement, increments `_refCount`.*
- `ULONG STDMETHODCALLTYPE` [Release](#) ()  
*One of the IUnknown methods we need to implement, decrements `_refCount`.*
- `HRESULT STDMETHODCALLTYPE` [QueryInterface](#) (REFIID riid, void \*\*ppInterface)  
*One of the IUnknown methods we need to implement, sets the appropriate interfaces.*
- `HRESULT STDMETHODCALLTYPE` [HandleAutomationEvent](#) (IUIAutomationElement \*pSender, EVENTID eventId)  
*The function that is called whenever we receive an event that the UIAutomation object registered for.*
- `HRESULT` [StartEventHandler](#) (HWND hDlg)  
*Send a message to start eventhandlers.*
- `void` [Uninitialize](#) ()  
*Here we tell the background thread to close down, then we cleanup.*

### Protected Member Functions

- void `cleanup` ()  
*Release object that where created on this thread.*
- HRESULT `registerEventHandler` ()  
*Register for all the event we want to register for.*
- void `removeEventHandler` ()  
*Removes all event handlers and stops listening.*

### Static Protected Member Functions

- static DWORD WINAPI `listenerThreadProc` (LPVOID lpParameter)  
*This is where the thread runs, listening for messages and take appropriate action.*

### Private Attributes

- LONG `_refCount`  
*Reference count.*
- IUIAutomation \* `automation`  
*UI Automation object we use the get the element tree.*
- IUIAutomationElement \* `rootElem`  
*Pointer to the root element in the UIA tree.*
- HWND `mainHwnd`  
*Pointer to the main window, can be used to send messages back (currently not used)*
- HANDLE `backThreadHandle`  
*Handle to the worker thread we are creating to listen for events.*
- DWORD `backThread`  
*ID to the worker thread we are creating to listen for events.*
- HANDLE `eventListenerReady`  
*Handle to the event object.*
- BOOL `eventHandlerAdded`  
*If event handlers has been added or not.*
- IUIAutomationCacheRequest \* `cache`  
*Cache for faster retrival of attributes.*
- int `_eventCount`  
*Event count.*

#### 2.8.1 Detailed Description

Responsible for registering for events and receiving these events.

None, except UIAutomation deals with this class, so unless you need to change what events are captured or what type of information you store, you don't have to worry about this class. This class receives the following events:

- Window Opend (WO)
- Element Invoked (EI)
- Menu Opened (MO)
- Text Changed (TC)

- Menu Mode Started (MMS) When the event occurs, this class mostly just sets what type of event it was, the rest is handled by the Events class. It also saves the following properties in cache:
  - Automation ID
  - Element type (control type)
  - The rectangle of each element
  - Element type name (localized control type)
  - Name
  - Process ID
  - Value (Specify)

1. Author Robin Stenvi - 2012-01-21

## 2.8.2 Constructor & Destructor Documentation

### 2.8.2.1 eventHandler::eventHandler ( )

Initilizes all variables to NULL and says that eventhandler has not been added yet.

1. Author Robin Stenvi

### 2.8.2.2 eventHandler::~~eventHandler ( )

Empty destructor.

1. Author Robin Stenvi

## 2.8.3 Member Function Documentation

### 2.8.3.1 DWORD WINAPI eventHandler::listenerThreadProc ( LPVOID *lpParameter* ) [static],[protected]

This is where the thread runs, listening for messages and take appropriate action.

This function should run on a separate thread. When the thread is created this function will set event-ListenerReady to say that it is ready. This is the thread that should register and remove the event handlers, otherwise something will fail.

1. Author Robin Stenvi

#### Parameters

<code>in</code>	<code>lpParameter</code>	Should point to an object of the current class.
-----------------	--------------------------	---

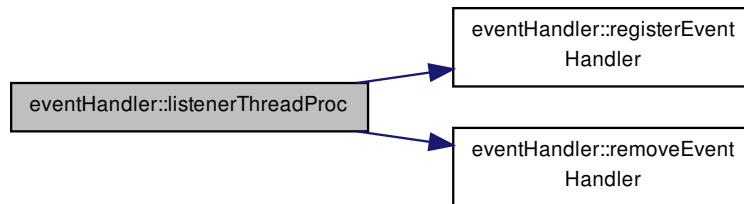
#### Returns

Return 1 if we fail to initialize COM library, 2 if we fail to say that we are ready and 0 if everything succeeds.

#### Remarks

If this function fails, it will just exit, and you will be unable to register for events, but the remaining code outside this function should still work.

Here is the call graph for this function:



### 2.8.3.2 void eventHandler::cleanup ( ) [protected]

Release object that where created on this thread.

#### Remarks

Should not be called directly, call `Uninitialize()`.

1. Author Robin Stenvi

### 2.8.3.3 HRESULT eventHandler::registerEventHandler ( ) [protected]

Register for all the event we want to register for.

1. Author Robin Stenvi

#### Returns

Returns `S_OK` if we succeed, if anythin else is returned, you should call `removeEventHandler()`.

#### Remarks

This function does NOT clean up after itself if we fail.

### 2.8.3.4 void eventHandler::removeEventHandler ( ) [protected]

Removes all event handlers and stops listening.

#### Remarks

This have to be done on the same thread in which it was created. Should only be called from `listenerThreadProc()`. Is safe to call even if events have not been added.

1. Author Robin Stenvi

### 2.8.3.5 ULONG STDMETHODCALLTYPE eventHandler::AddRef ( )

One of the `IUnknown` methods we need to implement, increments `_refCount`.

1. Author Robin Stenvi

#### Returns

InterlockedIncrement().

#### 2.8.3.6 ULONG STDMETHODCALLTYPE eventHandler::Release ( )

One of the IUnknown methods we need to implement, decrements \_refCount.

1. Author Robin Stenvi

#### Returns

Returns the current refCount, if the refCount is 0, this object has been deleted.

#### Remarks

This function deletes this object (when refCount == 0), don't call delete after this function.

#### 2.8.3.7 HRESULT STDMETHODCALLTYPE eventHandler::QueryInterface ( REFIID riid, void \*\* ppInterface )

One of the IUnknown methods we need to implement, sets the appropriate interfaces.

1. Author Robin Stenvi

#### Returns

Returns E\_NOINTERFACE or S\_OK.

Here is the call graph for this function:



#### 2.8.3.8 HRESULT STDMETHODCALLTYPE eventHandler::HandleAutomationEvent ( IUIAutomationElement \* pSender, EVENTID eventId )

The function that is called whenever we receive an event that the UIAutomation object registered for.

Whenever the the class need to handle additional events, this is the function you edit. Is a switch statement that identifies the events and uses Events class to get more information.

1. Author Robin Stenvi

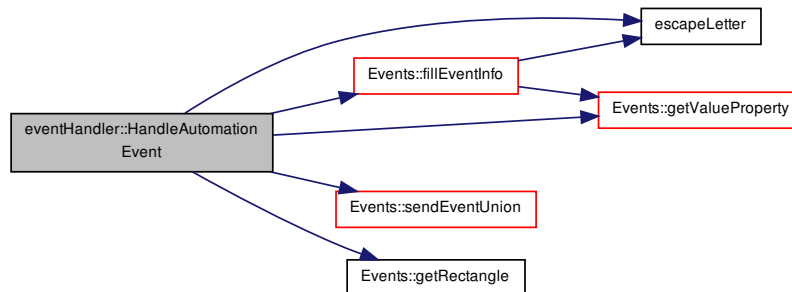
#### Parameters

in	<i>pSender</i>	The element that caused the event.
in	<i>eventId</i>	Says what type of event occurred.

**Returns**

Always returns S\_OK

Here is the call graph for this function:

**2.8.3.9 HRESULT eventHandler::StartEventHandler ( HWND hDlg )**

Send a message to start eventhandlers.

This function will start the new thread that starts all the necessary event handlers. The new thread will again register for the actual events, at this functions commands.

1. Author Robin Stenvi

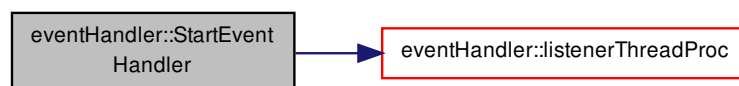
**Parameters**

in	<i>hDlg</i>	Should point to the main dialog window.
----	-------------	---

**Returns**

Returns S\_OK if we succeed, otherwise it returns the appropriate fail value from the Windows functions.

Here is the call graph for this function:

**2.8.3.10 void eventHandler::Uninitialize ( )**

Here we tell the background thread to close down, then we cleanup.



1. Author Robin Stenvi

#### Remarks

If we are unable to close the background thread, it will wait for 10 seconds, Windows will probably close it before then, but we avoid waiting infinitely if something fails.

Here is the call graph for this function:



## 2.9 eventInfoUnion Struct Reference

Contains the information we store about each UI event.

#### Public Attributes

- [UINT flag](#)  
*Flag to indicate event, used for faster processing.*
- [unionType type](#)  
*Which union is used.*
- `std::string` [description](#)  
*Description, to say what type of operation.*
- `std::string` [elemDescription](#)  
*Descriptive name of the element.*
- `std::string` [procName](#)  
*Process name.*
- `int` [elemType](#)  
*What type of element, like button.*
- `std::string` [elemID](#)  
*Automation ID of the element.*
- `DWORD` [time](#)  
*Timestamp with millisecond accuracy.*
- `std::string *` [desc](#)  
*Extra string for description.*
- `RECT *` [rect](#)  
*Provides a rectangle.*
- `double` [value](#)  
*A double value.*
- `int` [iValue](#)  
*An integer value, this will always operate as the flag.*

#### 2.9.1 Detailed Description

Contains the information we store about each UI event.

1. Author Robin Stenvi

## 2.10 Events Class Reference

Retrieves UI properties that we send to the server.

### Public Member Functions

- [Events](#) ()  
*Constrictor to initialize all variables.*
- [~Events](#) ()  
*Destructor to delete all variables.*
- `std::string` [getProcName](#) (IUIAutomationElement \*pSender, DWORD proclIn=0)  
*Returns the process executables name, based on UIA element.*
- `std::string` [getElemDescription](#) (IUIAutomationElement \*pSender)  
*Gets an elements name.*
- `std::string *` [getValueProperty](#) (IUIAutomationElement \*pSender)  
*Returns the value of an element.*
- `RECT *` [getRectangle](#) (IUIAutomationElement \*pSender)  
*Returns the rectangle of an element.*
- `int` [getControlType](#) (IUIAutomationElement \*pSender)  
*Returns what type of element it is, like "button".*
- `std::string` [getElemId](#) (IUIAutomationElement \*pSender)  
*Get the automation ID of the element.*
- `void` [removeId](#) (IUIAutomationElement \*pSender, DWORD proclIn=0)  
*Removes a process ID from our list, and sorts afterwards.*
- `bool` [sendEventUnion](#) ([eventInfoUnion](#) \*info)  
*Send an event to the server.*
- `bool` [fillEventInfo](#) ([eventInfoUnion](#) \*info, IUIAutomationElement \*pSender, `bool` gatherType=true)  
*Fills all the available after an UI Automation event has happened.*
- `bool` [isEdit](#) (IUIAutomationElement \*pSender)  
*Checks if the element is an edit field or not.*
- `bool` [isDocument](#) (IUIAutomationElement \*pSender)  
*Checks if the element is a document field or not.*
- `void` [deleteEventUnion](#) ([eventInfoUnion](#) \*info)  
*Deletes all the elements in an eventInfoUnion.*

### Private Attributes

- `HRESULT` [hr](#)  
*Permanent variable we use to check if the function call failed.*
- `processList` [list](#) [`MAX`]  
*Stores all the processes and IDs for fast retrieval.*
- `UINT` [counter](#)  
*The current place in our array.*

### Static Private Attributes

- static const int `MAX_LEN` = 100  
*The maximum length we allow for text, if we are going to send it to the server.*
- static const UINT `MAX` = 1000  
*Max size of the array.*

#### 2.10.1 Detailed Description

Retrieves UI properties that we send to the server.

1. Author Robin Stenvi

#### 2.10.2 Constructor & Destructor Documentation

##### 2.10.2.1 Events::Events ( )

Constrictor to initialize all variables.

1. Author Robin Stenvi

##### 2.10.2.2 Events::~~Events ( )

Destructor to delete all variables.

1. Author Robin Stenvi

#### 2.10.3 Member Function Documentation

##### 2.10.3.1 std::string Events::getProcName ( IUIAutomationElement \* *pSender*, DWORD *procldn* = 0 )

Returns the process executables name, based on UIA element.

It first tries to find the element in our list, if it's not found, it tries to get it. Returns |unknown| if it couldn't be found. If it was found, it inserts it in our list, sorts the list, then returns the name. You need either UI Automation element or prcess ID to use this function.

1. Author Robin Stenvi

#### Parameters

in	<i>pSender</i>	The UI Automation element
in	<i>procldn</i>	The process ID, if we don't have an UI Automation element, default value is 0.

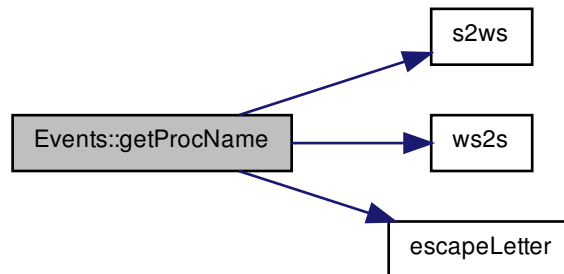
#### Remarks

Elements are never thrown out of the list. This is not desirable, but not a huge problem, since there is a limit to how many different programs the user can use. This function does not validate *pSender*.

**Returns**

Returns the process name in the form of `std::string`, or `UnknownElem` if we are unable to find it.

Here is the call graph for this function:



### 2.10.3.2 `std::string Events::getElemDescription ( IUIAutomationElement * pSender )`

Gets an elements name.

1. Author Robin Stenvi

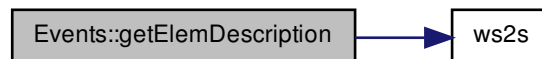
**Parameters**

<code>in</code>	<code>pSender</code>	The UI Automation element
-----------------	----------------------	---------------------------

**Returns**

Returns `UnknownElem` if it can't find it, `TooLongElem` if it is deemed too long, `EmptyElem` if it's empty. Should never return an empty string

Here is the call graph for this function:



### 2.10.3.3 `std::string * Events::getValueProperty ( IUIAutomationElement * pSender )`

Returns the value of an element.

This property should be what the user sees, when looking at the element, it returns the name, `|Too long|`, or `"-"` if empty. The last one should be compatible with the syslog protocol. Should never return `NULL`.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>pSender</i>	The UI Automation element
-----------	----------------	---------------------------

#### Returns

Returns the value. TooLongElem if it's too long, EmptyElem if it's empty and UnknownElem if we can't find the value.

Here is the call graph for this function:



#### 2.10.3.4 RECT \* Events::getRectangle ( IUIAutomationElement \* *pSender* )

Returns the rectangle of an element.

Return all -1 if the rectangle could not be found, should never return NULL.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>pSender</i>	The UI Automation element
-----------	----------------	---------------------------

#### Returns

Returns the rectangle, allocated with new

#### 2.10.3.5 int Events::getControlType ( IUIAutomationElement \* *pSender* )

Returns what type of element it is, like "button".

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>pSender</i>	The UI Automation element
-----------	----------------	---------------------------

**Returns**

An int value indicating what type of element it is.

**2.10.3.6 std::string Events::getElemId ( IUIAutomationElement \* *pSender* )**

Get the automation ID of the element.

Returns UnknownElem if it could not be found or it is empty, this is compatible with Syslog. If it returns a real element, it should be unique among it's sibling and it should not change the next time the program starts up. It might change hen a new version of the program comes out.

1. Author Robin Stenvi

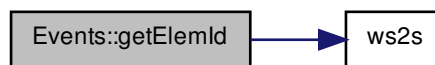
**Parameters**

<i>in</i>	<i>pSender</i>	The UI Automation element
-----------	----------------	---------------------------

**Returns**

Returns the element, returns EmptyElem if it's empty, UnknownElem if we are unable to find it.

Here is the call graph for this function:

**2.10.3.7 void Events::removeId ( IUIAutomationElement \* *pSender*, DWORD *proclIn* = 0 )**

Removes a process ID from our list, and sorts afterwards.

You only need to supply one of the values.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>pSender</i>	Element of the process to be removed, use NULL if you use <i>proclIn</i> instead
<i>in</i>	<i>proclIn</i>	Process ID that should be removed, default is 0

**Remarks**

This Function does not validate *pSender*

**2.10.3.8 bool Events::sendEventUnion ( eventInfoUnion \* *info* )**

Send an event to the server.

1. Author Robin Stenvi

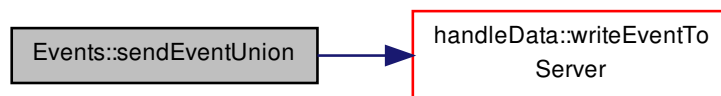
#### Parameters

<i>in</i>	<i>info</i>	Information about the event that should be sent to the server.
-----------	-------------	--

#### Returns

Returns the value from `handleData::writeEventToServer()`

Here is the call graph for this function:



**2.10.3.9** `bool Events::fillEventInfo ( eventInfoUnion * info, UIAutomationElement * pSender, bool gatherType = true )`

Fills all the available after an UI Automation event has happened.

1. Author Robin Stenvi

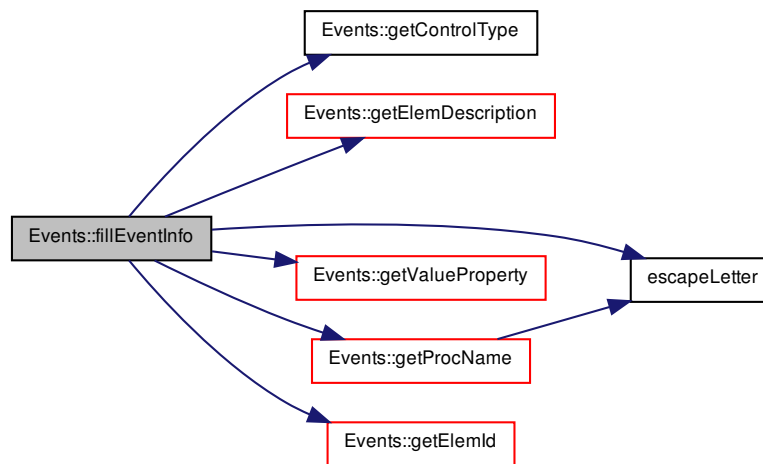
#### Parameters

<i>in, out</i>	<i>info</i>	The struct that should be filled with information.
<i>in</i>	<i>pSender</i>	The UI Automation element
<i>in</i>	<i>gatherType</i>	Default value is true, if false, we should not gather controltype, which means that is has already been set.

#### Returns

Returns true if we are successful, otherwise false.

Here is the call graph for this function:



#### 2.10.3.10 `bool Events::isEdit ( IUIAutomationElement * pSender )`

Checks if the element is an edit field or not.

1. Author Robin Stenvi

##### Parameters

<i>in</i>	<i>pSender</i>	The UI Automation element
-----------	----------------	---------------------------

##### Returns

Returns true if this is an edit field, false if it's not or we are unable to check

#### 2.10.3.11 `bool Events::isDocument ( IUIAutomationElement * pSender )`

Checks if the element is a document field or not.

1. Author Robin Stenvi

##### Parameters

<i>in</i>	<i>pSender</i>	The UI Automation element
-----------	----------------	---------------------------

##### Returns

Returns true if it is a document field.

#### 2.10.3.12 `void Events::deleteEventUnion ( eventInfoUnion * info )`

Deletes all the elements in an eventInfoUnion.



Check every element if it's NULL. Should always call this function for delete. If you delete individually, you should set that variable to NULL.

1. Author Robin Stenvi

#### Parameters

in, out	<i>info</i>	The struct that should be deleted.
---------	-------------	------------------------------------

## 2.11 sendData::Excluded Struct Reference

Holds times that are excluded from the user.

### 2.11.1 Detailed Description

Holds times that are excluded from the user.

## 2.12 sendData::ExcludeIndex Struct Reference

Which index the timestamps points to.

### 2.12.1 Detailed Description

Which index the timestamps points to.

## 2.13 sendData::File Struct Reference

Holds all information we need to know about a file, is filled gradually.

#### Public Attributes

- std::wstring [file](#)  
*Full path to the file.*
- SYSTEMTIME [timeStart](#)  
*When the session started.*
- SYSTEMTIME [timeStop](#)  
*When the session ended.*
- int [length](#)  
*How many events it contains.*
- int [currLine](#)  
*Which line we are going to start at, default should be 0.*
- std::vector< std::wstring > [allLines](#)  
*All the events.*
- std::vector< [Excluded](#) > [excluded](#)  
*List of excluded timestamps.*
- std::vector< [ExcludeIndex](#) > [exIndex](#)  
*List of excluded indexes.*

### 2.13.1 Detailed Description

Holds all information we need to know about a file, is filled gradually.

## 2.14 filterSettings Class Reference

Handles all the user settings for filtering data to screen.

### Public Member Functions

- [filterSettings](#) (CWnd \*pParent=NULL)  
*Constructor, should just set current filter to 0 or saved filter.*
- `afx_msg void mouseAll ()`  
*Checks or unchecks all mouse events.*
- `afx_msg void mouseMove ()`  
*Checks or unchecks mouse move.*
- `afx_msg void mouseWheel ()`  
*Checks or unchecks mouse wheel.*
- `afx_msg void mousePress ()`  
*Checks or unchecks all mouse presses.*
- `afx_msg void mousePressUp ()`  
*Checks or unchecks mouse presses up.*
- `afx_msg void mousePressDown ()`  
*Checks or unchecks mouse releases.*
- `afx_msg void mousePressLeft ()`  
*Checks or unchecks left mouse button.*
- `afx_msg void mousePressMiddle ()`  
*Checks or unchecks middle mouse button.*
- `afx_msg void mousePressRight ()`  
*Checks or unchecks right mouse button.*
- `afx_msg void keyPressAll ()`  
*Checks or unchecks all key events.*
- `afx_msg void keyPressUp ()`  
*Checks or unchecks key release events.*
- `afx_msg void keyPressDown ()`  
*Checks or unchecks key down events.*
- `virtual INT_PTR DoModal ()`  
*Initiates modal for the dialog.*
- `void setBoxes ()`  
*Is called immediately when the box appear to set the appropriate boxes, which was last set.*
- `UINT retFilter ()`  
*Returns the current filter settings.*
- `afx_msg void clearButton ()`  
*Unchecks all boxes.*
- `afx_msg void MarkAllButton ()`  
*Checks all boxes.*

### Private Attributes

- CButton [KPDown](#)  
*Checkbox for key press down.*
- CButton [KPUp](#)  
*Checkbox for key press up.*
- CButton [MPRight](#)  
*Checkbox for mouse press right button.*
- CButton [MPMiddle](#)  
*Checkbox for mouse press middle button.*
- CButton [MPLeft](#)  
*Checkbox for mouse press left button.*
- CButton [MPDown](#)  
*Checkbox for mouse press down.*
- CButton [MPUp](#)  
*Checkbox for mouse press up.*
- CButton [MP](#)  
*Checkbox for mouse press all.*
- CButton [MW](#)  
*Checkbox for mouse wheel.*
- CButton [MV](#)  
*Checkbox for mouse move.*
- CButton [MA](#)  
*Checkbox for mouse all.*
- CButton [KPA](#)  
*Checkbox for key press all.*
- UINT [filter](#)  
*The current filter settings.*

#### 2.14.1 Detailed Description

Handles all the user settings for filtering data to screen.

1. Author Robin Stenvi

#### 2.14.2 Constructor & Destructor Documentation

##### 2.14.2.1 filterSettings::filterSettings ( CWnd \* *pParent* = NULL )

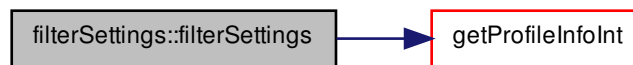
Constructor, should just set current filter to 0 or saved filter.

1. Author Robin Stenvi

### Parameters

in	<i>pParent</i>	Sent to parent.
----	----------------	-----------------

Here is the call graph for this function:



### 2.14.3 Member Function Documentation

#### 2.14.3.1 void filterSettings::mouseAll ( )

Checks or unchecks all mouse events.

1. Author Robin Stenvi

#### 2.14.3.2 void filterSettings::mouseMove ( )

Checks or unchecks mouse move.

1. Author Robin Stenvi

#### 2.14.3.3 void filterSettings::mouseWheel ( )

Checks or unchecks mouse wheel.

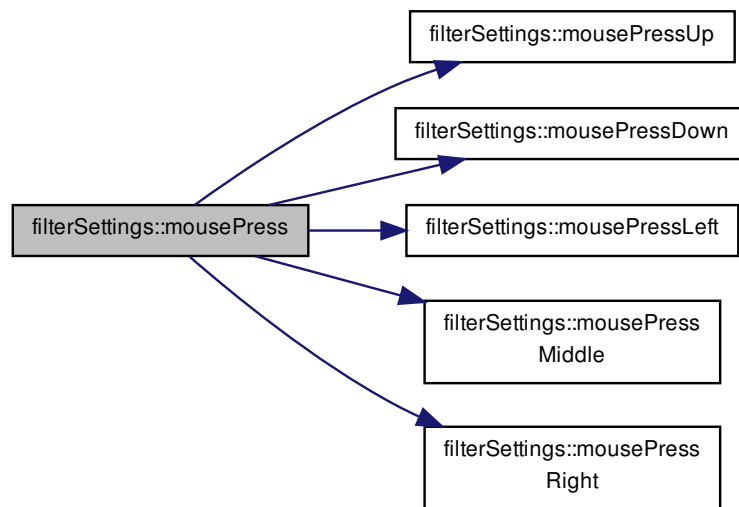
1. Author Robin Stenvi

#### 2.14.3.4 void filterSettings::mousePress ( )

Checks or unchecks all mouse presses.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.14.3.5 void filterSettings::mousePressUp ( )

Checks or unchecks mouse presses up.

1. Author Robin Stenvi

##### Remarks

This include left/right/middle, but doesn't mark it.

#### 2.14.3.6 void filterSettings::mousePressDown ( )

Checks or unchecks mouse releases.

1. Author Robin Stenvi

##### Remarks

This include left/right/middle, but doesn't mark it.

#### 2.14.3.7 void filterSettings::mousePressLeft ( )

Checks or unchecks left mouse button.

1. Author Robin Stenvi

##### Remarks

This include press and release, but doesn't mark it.

**2.14.3.8 void filterSettings::mousePressMiddle ( )**

Checks or unchecks middle mouse button.

1. Author Robin Stenvi

**Remarks**

This include press and release, but doesn't mark it.

**2.14.3.9 void filterSettings::mousePressRight ( )**

Checks or unchecks right mouse button.

1. Author Robin Stenvi

**Remarks**

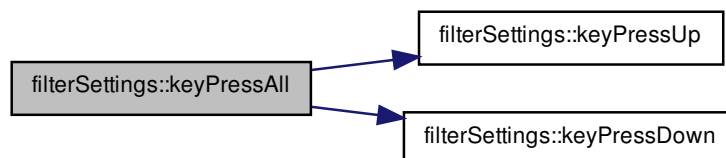
This include press and release, but doesn't mark it.

**2.14.3.10 void filterSettings::keyPressAll ( )**

Checks or unchecks all key events.

1. Author Robin Stenvi

Here is the call graph for this function:

**2.14.3.11 void filterSettings::keyPressUp ( )**

Checks or unchecks key release events.

1. Author Robin Stenvi

**2.14.3.12 void filterSettings::keyPressDown ( )**

Checks or unchecks key down events.

1. Author Robin Stenvi

#### 2.14.3.13 INT\_PTR filterSettings::DoModal ( ) [virtual]

Initiates modal for the dialog.

1. Author Robin Stenvi

#### Returns

CDialog::DoModal().

#### 2.14.3.14 void filterSettings::setBoxes ( )

Is called immediately when the box appear to set the appropriate boxes, which was last set.

1. Author Robin Stenvi

#### 2.14.3.15 UINT filterSettings::retFilter ( )

Returns the current filter settings.

1. Author Robin Stenvi

#### Returns

Returns the filter.

#### 2.14.3.16 void filterSettings::clearButton ( )

Unchecks all boxes.

1. Author Robin Stenvi

Here is the call graph for this function:

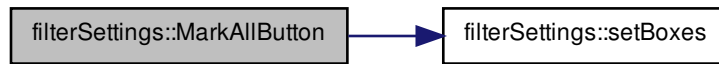


#### 2.14.3.17 void filterSettings::MarkAllButton ( )

Checks all boxes.

1. Author Robin Stenvi

Here is the call graph for this function:



## 2.15 focusEventHandler Class Reference

Handles focus change events.

### Public Member Functions

- [focusEventHandler](#) ()
- ULONG STDMETHODCALLTYPE [AddRef](#) ()
- ULONG STDMETHODCALLTYPE [Release](#) ()
- HRESULT STDMETHODCALLTYPE [QueryInterface](#) (REFIID riid, void \*\*ppInterface)
- HRESULT STDMETHODCALLTYPE [HandleFocusChangedEvent](#) (UIAutomationElement \*pSender)
  - Handles events and set the password field if necessary, also detect screen changes.*
- HRESULT [StartEventHandler](#) (HWND hDlg)
  - Start the actual new thread that will see new UI Automation events.*
- void [Uninitialize](#) ()
  - Here we tell the background thread to close down.*
- void [removeEventHandler](#) ()
  - Removes all event handlers and stops listening.*
- bool [checkScreen](#) ()
  - Checks which screen is currently active and send a message to server if it has changed.*

### Protected Member Functions

- void [cleanup](#) ()
  - Release object that where created on this thread.*
- HRESULT [registerEventHandler](#) ()
  - Registers which events we are interested in and builds up the cache.*

### Static Protected Member Functions

- static DWORD WINAPI [listenerThreadProc](#) (\_\_in LPVOID lpParameter)
  - This is where the thread runs, listening for messages and take appropriate action.*



### Private Attributes

- LONG `_refCount`  
*Reference count.*
- IUIAutomation \* `automation`  
*UI Automation object we use to get the element tree.*
- IUIAutomationElement \* `rootElem`  
*Pointer to the root element in the UIA tree.*
- HWND `mainHwnd`  
*Pointer to the main window, can be used to send messages back (currently not used)*
- HANDLE `backThreadHandle`  
*Handle to the worker thread we are creating to listen for events.*
- DWORD `backThread`  
*ID to the worker thread we are creating to listen for events.*
- HANDLE `eventListenerReady`  
*Handle to the event object.*
- BOOL `eventHandlerAdded`  
*If event handlers have been added or not.*
- BOOL `pass`  
*If the current element is a password field or not.*
- IUIAutomationCacheRequest \* `cache`  
*Cache for faster retrieval of attributes.*
- HMONITOR `last`  
*The last monitor we saw, used to see changes in monitor.*

#### 2.15.1 Detailed Description

Handles focus change events.

This class handles focus change events, this is usually when the user changes the foreground window. In the cache we store the following properties:

- Automation ID
- Element type (control type)
- The rectangle of each element
- If it is a password or not
- Element type name (localized control type)
- Name
- Process ID

1. Author Robin Stenvi - 2012-01-21 - 2012-01-21 (Last modified)

#### 2.15.2 Constructor & Destructor Documentation

##### 2.15.2.1 focusEventHandler::focusEventHandler ( )

1. Author Robin Stenvi

### 2.15.3 Member Function Documentation

#### 2.15.3.1 DWORD WINAPI focusEventHandler::listenerThreadProc ( ..in LPVOID *lpParameter* ) [static], [protected]

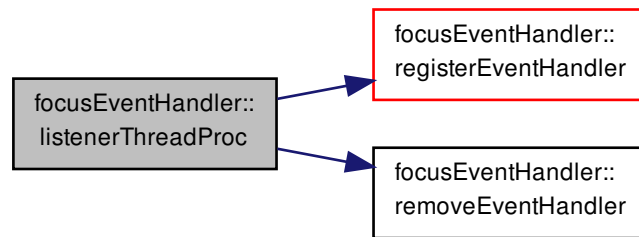
This is where the thread runs, listening for messages and take appropriate action.

1. Author Robin Stenvi

#### Parameters

in	<i>lpParameter</i>	Should point to an object of the current class.
----	--------------------	---

Here is the call graph for this function:



#### 2.15.3.2 void focusEventHandler::cleanup ( ) [protected]

Release object that where created on this thread.

We must not remove the event handler here, it has to be removed on the same thread that added it.

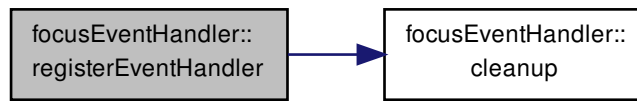
1. Author Robin Stenvi

#### 2.15.3.3 HRESULT focusEventHandler::registerEventHandler ( ) [protected]

Registers which events we are interested in and builds up the cache.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.15.3.4 ULONG STDMETHODCALLTYPE focusEventHandler::AddRef ( )

1. Author Robin Stenvi

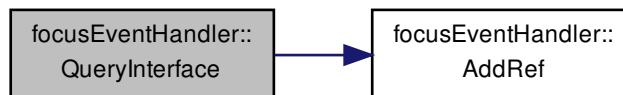
#### 2.15.3.5 ULONG STDMETHODCALLTYPE focusEventHandler::Release ( )

1. Author Robin Stenvi

#### 2.15.3.6 HRESULT STDMETHODCALLTYPE focusEventHandler::QueryInterface ( REFIID riid, void \*\* ppInterface )

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.15.3.7 HRESULT STDMETHODCALLTYPE focusEventHandler::HandleFocusChangedEvent ( IUIAutomationElement \* pSender )

Handles events and set the password field if necessary, also detect screen changes.

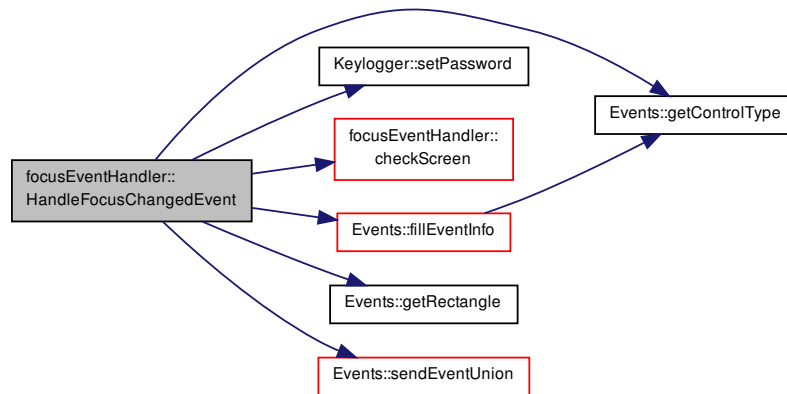
We only check the password field if we see an edit field.

1. Author Robin Stenvi

## Remarks

Checks change in screen all the time, this might be a bit time-consuming. Might be able to keep a list of which application are on which screens and then only check for new applications, but then we have to follow visual change and keep the list at an element basis, since an application can be on multiple screens.

Here is the call graph for this function:



## 2.15.3.8 HRESULT focusEventHandler::StartEventHandler ( HWND hDlg )

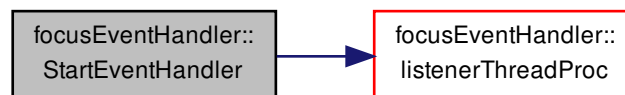
Start the actual new thread that will see new UI Automation events.

1. Author Robin Stenvi

## Parameters

<code>in</code>	<code>hDlg</code>	Should point to the main dialog window
-----------------	-------------------	--

Here is the call graph for this function:

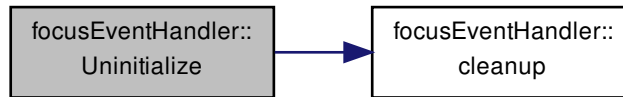


## 2.15.3.9 void focusEventHandler::Uninitialize ( )

Here we tell the background thread to close down.

1. Author Robin Stenvi

Here is the call graph for this function:



### 2.15.3.10 void focusEventHandler::removeEventHandler ( )

Removes all event handlers and stops listening.

1. Author Robin Stenvi

#### Remarks

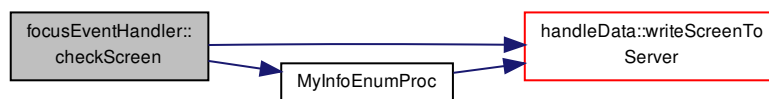
This have to be done on the same thread in which it was created

### 2.15.3.11 bool focusEventHandler::checkScreen ( )

Checks which screen is currently active and send a message to server if it has changed.

1. Author Robin Stenvi

Here is the call graph for this function:



## 2.16 formatData Class Reference

Retrieves the real time that an event happened at and gives it as a readable string.

#### Public Member Functions

- [formatData \( \)](#)  
*Retrives the current status regarding, how long since the system was booted.*
- `char * timeNow` (DWORD time)  
*Gives a string with the real date and time given in the parameter.*
- `std::string timeNowStd` (DWORD time)  
*Retrieves a string with the correct time, in respect to time.*

### Protected Member Functions

- SYSTEMTIME `getRealTime` (DWORD *ts*)  
*Get the actual time when an event occurred.*

### Private Attributes

- SYSTEMTIME `startUp`  
*Get time when we started the application.*
- DWORD `millStartUp`  
*Get number of milliseconds in respect to the previous time.*
- FILETIME `ft`  
*Holds the filetime for the last time we saw.*
- DWORDLONG `millStartUp64`  
*Number of milliseconds since the system was booted.*
- int `mult`  
*Specify the amount of 49 days wrap-around we have.*
- HANDLE `mutex`  
*The mutex that we lock when we are changing data.*

#### 2.16.1 Detailed Description

Retrieves the real time that an event happened at and gives it as a readable string.

This class also make sure that we don't wrap around if the computer has been up for more than 49 days, which is all that a DWORD can handle.

1. Author Robin Stenvi

#### 2.16.2 Constructor & Destructor Documentation

##### 2.16.2.1 `formatData::formatData ( )`

Retrives the current status regarding, how long since the system was booted.

This information is used later to detect if the system has been up for longer than 49 days.

1. Author Robin Stenvi

#### 2.16.3 Member Function Documentation

##### 2.16.3.1 SYSTEMTIME `formatData::getRealTime ( DWORD ts )` `[protected]`

Get the actual time when an event occurred.

1. Author Robin Stenvi

#### Parameters

<code>in</code>	<code>ts</code>	Milliseconds since the system started, that the event happened
-----------------	-----------------	--

**Returns**

The real time that the parameter represent.

**Remarks**

Will use a mutex to verify that no other thread is working on the time objects

**2.16.3.2 char \* formatData::timeNow ( DWORD *time* )**

Gives a string with the real date and time given in the parameter.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>time</i>	Number of milliseconds since the system was started
-----------	-------------	---

**Returns**

Returns a date and time with the format: YYYY-MM-DDTHH:MM:SS.mmmm

Here is the call graph for this function:

**2.16.3.3 std::string formatData::timeNowStd ( DWORD *time* )**

Retrieves a string with the correct time, in respect to time.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>time</i>	Number of milliseconds since boot time that the event happened
-----------	-------------	--

**Returns**

Returns a string that is compatible with the syslog protocol

Here is the call graph for this function:

**2.17 handleData Class Reference**

In charge of writing all the data to the server, also does some filtering.

**Classes**

- struct [lastAll](#)  
*Holds all the previous events, is used to find which events correlate to other events.*

**Public Member Functions**

- [handleData](#) ()  
*Initiate all variables and create transmission object.*
- [~handleData](#) ()  
*Need to delete everything it has used and flush all remaining output, if any.*
- char \* [writeTime](#) (const char \*sentence, int eventType)  
*Write time and a message to server, used at pause/resume/stop/start.*
- bool [writeEventToServer](#) ([eventInfoUnion](#) \*out)  
*Takes an UI AUtomation event and writes it to the server, GUI and file if necessary.*
- bool [writeMouseToServer](#) ([MouseInfo](#) out)  
*Takes a mouse event and writes it to the server, GUI and file if necessary.*
- bool [writeKeyToServer](#) ([KeyInfo](#) out)  
*Takes a key event and writes it to the server, GUI and file if necessary.*
- bool [writeHWToServer](#) ([sysResources](#) \*HW)  
*Takes a hardware event and writes it to the server, GUI and file if necessary.*
- bool [writeDevToServer](#) ([deviceInfo](#) dev)  
*Takes a device event and writes it to the server, GUI and file if necessary.*
- bool [writeScreenToServer](#) ([Screen](#) screen)  
*Takes a screen event and writes it to the server, GUI and file if necessary.*
- bool [writeHIDToServer](#) ([HIDDevice](#) device)  
*Takes an input device event and writes it to the server, GUI and file if necessary.*
- bool [writeKeyboardToServer](#) ([KeyboardDevice](#) kd)  
*Writes information about the keyboard to the server.*
- bool [writeStringToServer](#) (std::string str)  
*Takes whatever string it gets and write it to the server.*



- void [togglePaused](#) ()  
*Toggles the pause value, just set it as the opposite.*
- bool [retPaused](#) ()  
*Returns the current value of the pause variable.*
- void [updateSID](#) ()  
*Update and set the session number.*
- void [setLogPort](#) (int port)  
*Changes the port number we should use.*
- void [setLogAddr](#) (std::wstring addr)  
*Changes the server adress that we should use.*
- int [retIntConnection](#) ()  
*Returns the current connection status.*
- bool [initCom](#) ()  
*Initiate the COM library, useful if you start a new thread on a static function.*
- void [setFilter](#) (int n)  
*Change the current filter settings for what to display to the user.*

#### Static Public Member Functions

- static void [toggleServerStatic](#) (void \*p)  
*Static helper function, so we can call toggleServer().*
- static void [sendListToServerStatic](#) (void \*p)  
*Static function you can use when starting a new thread, function sends the list we kept when server connection was unable.*
- static void [sendFullListToServerStatic](#) (void \*p)  
*Sends completeList to server, static functions that can be used when starting a new thread.*

#### Protected Member Functions

- char \* [getTimestamp](#) (const char \*from)  
*Retrives timestamp from syslog string in a more readable format.*
- std::string [getTimestamp](#) (std::string from)  
*Retreives the timestamp from a syslog event string.*
- std::string [getTimestamp](#) (bool date=false)  
*Retrieves a timestamp with seconds accuracy.*
- std::string [getCsvMouse](#) ([MouseInfo](#) input, int ev, int rel=0)  
*Retrives a mouse event in CSV format, can be used if you need to write local storage.*
- std::string [getCsvUIA](#) ([eventInfoUnion](#) input, int ev, int rel=0)  
*Retrives a software event in CSV format, can be used if you need to write local storage.*
- std::string [getCsvKey](#) ([KeyInfo](#) input, int ev, int rel=0)  
*Retrives a key event in CSV format, can be used if you need to write local storage.*
- std::string [getCsvHW](#) ([sysResources](#) \*HW)  
*Retrives a system resource event in CSV format, can be used if you need to write local storage.*
- std::string [getCsvDev](#) ([deviceInfo](#) dev)  
*Retrives a device insert/remove event in CSV format, can be used if you need to write local storage.*
- std::string [getCsvScreen](#) ([Screen](#) screen)  
*Retrieves CSV format of a pysical screen change event.*
- std::string [getCsvHID](#) ([HIDDevice](#) device)  
*Retrieves CSV version of a HID change.*
- std::string [getCsvKeyboard](#) ([KeyboardDevice](#) kd)

- Retrieves CSV format with informatio about a physical keyboard.*

  - std::string `getCsvBelt` (int evType, DWORD time)

*Retreives a string in CSV format for the start/stop/pause/resume messages.*
- std::string `getCsvRectangle` (RECT \*rect)

*Retrives a rectangle in CSV format, can be used if you need to write local storage.*
- std::string `getFormatMouse` (MouseInfo input, int rel=0)

*Retrieves the complete structured data that should be sent in any mouse event.*
- std::string `getFormatUIA` (eventInfoUnion input, int rel=0)

*Gets the structured data of a software event.*
- std::string `getFormatKey` (KeyInfo input, int rel=0)

*Gets the structured data of a key event.*
- std::string `getFormatHW` (sysResources \*HW)

*Gets the structured data of a HW average event.*
- std::string `getFormatKeyboard` (KeyboardDevice kd)

*Retrieves the structured data with informatio about a physical keyboard.*
- std::string `getFormatDev` (deviceInfo dev)

*inline function to get structured data from a device event*
- std::string `getFormatScreen` (Screen screen)

*Gets the structured data of a screen event.*
- std::string `getFormatHID` (HIDDevice device)

*Gets the structured data of a input device event.*
- std::string `getFormatRectangle` (RECT \*rect)

*Returns retctangle in the form of structured data that can be sent to the server.*
- std::string `getEventToServer` (eventInfoUnion \*out, int backRef, int ev=0)

*Retrieves a syslog compatible message corresponding to a software event.*
- std::string `getMouseToServer` (MouseInfo out, int backRef, int ev=0)

*Retrieves a syslog compatible message corresponding to a mouse event.*
- std::string `getKeyToServer` (KeyInfo out, int backRef, int ev=0)

*Retrieves a syslog compatible message corresponding to a key event.*
- std::string `getHWToServer` (sysResources \*HW, int ev=0)

*Retrieves a syslog compatible message corresponding to a system resource usage.*
- std::string `getDevToServer` (deviceInfo dev, int ev=0)

*Retrieves a syslog compatible message corresponding to a device insert /remove.*
- std::string `getScreenToServer` (Screen screen, int ev=0)

*Retrieves a syslog compatible message corresponding to a physical screen change event.*
- std::string `getHIDToServer` (HIDDevice device, int ev=0)

*Retrieves a syslog compatible message corresponding to a physical device.*
- std::string `getKeyboardToServer` (KeyboardDevice kd, int ev=0)

*Retrieves a syslog compatible message corresponding to information about a keyboard.*
- std::string `getBeltToServer` (int evType, DWORD time)

*Get a syslog message corresponding to a start/stop/pause/resume in belt event.*
- std::string `getDescSentenceMouse` (MouseInfo input, std::string finalTime)

*Gets a descriptive sentence for mouse events that can be displayed to the user.*
- std::string `getDescSentenceKey` (KeyInfo input, std::string finalTime)

*Gets a descriptive sentence for key events that can be displayed to the user.*
- void `sendToServer` (std::string str)

*Send a single event to the server, the string should already be formatted correctly.*
- void `sendListToServer` ()

*Send all that is stored temporarily and send it to the server.*
- void `sendData` (std::string data, UINT type=0)

*Used to send string back to the GUI, it makes the decision whether it should be displayed or not.*

- BOOL `writeData` (std::string data)  
*Writes whatever data that comes in to file.*
- void `sendFullListToServer` ()  
*Sends everything we have stored temporarily in a list to the server.*
- void `writeMissing` ()  
*Should write any files that we didn't have time to send, for whatever reason, should also mark it in the profile file.*
- void `sendMissing` ()  
*Should send any files that we were unable to send the last time we ran the program.*
- void `writeAll` (std::string str)  
*Writes an event to file or server.*
- bool `startNewSession` (std::string bufTmp)  
*Initializes all the variables needed to start a session.*
- void `stopCurrentSession` ()  
*Closes a running session.*
- void `accessKey` (KeyInfo out)  
*If the access key for pause (`{alt|+|pause|}`) has been hit we tell the GUI that logging should be paused.*
- void `toggleServer` ()  
*Start or stop the connection to the server, depending on what the current status is.*

#### Static Protected Member Functions

- static void `writeMissingStatic` (void \*p)  
*Wrapper to sendMissing(), used when calling on a thread, will check if Internet is available every 5 seconds and start sending as soon as Internet is available.*

#### Private Attributes

- std::vector< std::string > `completeList`  
*Here we store all the events before sending them to the server when we have gathers MAX\_FULL\_LIST events.*
- std::vector< std::string > `serverList`  
*All the events we have not sent to the server yet, they are held here temporarily because the connection to the server is down for whatever reason.*
- Syslog1 \* `syslog`  
*Object that handles all logs to the server.*
- std::wstring `fileName`  
*The filename that we are currently writing to.*
- CStdioFile `file`  
*The file that we are writing to.*
- bool `fileOpened`  
*If we have an active file or not.*
- bool `paused`  
*The application has paused and should not send events.*
- int `serverStarted`  
*If we have a valid connection to the server or not.*
- char \* `sessionID`  
*Which session we are on.*
- int `events`  
*Number of events in this session.*
- int `storageType`

- The current method of storage.*
- bool `running`
  - Has the logging been started or stopped.*
- int `currFilter`
  - What does the user want to display to the screen.*
- `lastAll` `lasts`
  - Record of the last events in certain categories.*
- `eventInfoUnion` \* `last`
  - Record of the last Software event sent, used for filtering equal events.*
- RECT \* `lastRect`
  - Last active area we have seen, used to see where mouse presses went We don't need to interpret this, so it would maybe be more efficient to just have a `std::string`, especially when we don't write a local copy.*
- DWORD `dwThreadList`
  - ID for thread that sends `serverList` to server.*
- HANDLE `threadListH`
  - Handle for thread that sends `serverList` to server.*
- DWORD `firstThreadID`
  - ID to thread that send the previous session events to server.*
- HANDLE `firstThreadH`
  - Handle to thread that send the previous session events to server.*
- DWORD `dwThreadSendFull`
  - ID to the thread sending `completeList` to the server.*
- HANDLE `dwThreadSendFullH`
  - Handle to the thread sending `completeList` to the server.*
- CStdioFile `localStorageFile`
  - The file we use for raw output.*
- `std::wstring` `localStorageFileName`
  - The filename for our raw output file.*
- bool `localFileOpened`
  - If the file is opened.*
- `std::wstring` `localFolder`
  - The path to our local file for raw storage.*
- HANDLE `rawOutputM`
  - Mutex for our file to raw output.*
- HANDLE `CSVFileM`
  - Mutex for the CSV file.*
- HANDLE `elemListM`
  - Mutex for list of elements, if we store a certain amount before sending to server.*
- HANDLE `createThreadM`
  - Mutex so we don't create multiple threads on same dataset.*
- HANDLE `lastDataM`
  - Mutex for `lastAll` struct.*
- CWnd \* `mainDialog`
  - Handle to the main dialog, used to post messages.*

#### Static Private Attributes

- static const int `MAX_KEYS` = 10
  - Max number of last keys we can hold.*
- static const int `MAX_FULL_LIST` = 500
  - Max number of events we store temporarily before sending it to the server is used to limit the number of connections that need to be made.*
- static const int `MAX_STORAGE` = 10000
  - Number of elements we can store locally before sending to server, each event is about 100B.*

### 2.17.1 Detailed Description

In charge of writing all the data to the server, also does some filtering.

This is the single point which all data must pass through to reach the server. All events come in as structs representing the data, if you need more data in the future, you need to change the struct and change how this class handles the struct. On some software events it filter equal events, but mostly it just makes it into a syslog compatible string and send it to the server. This class also handles the pause possibility, if the program is paused, all action will be sent here, but will not be sent to the server and will not be stored locally. This class also correlates previous events by keeping a record of what was the last thing that occurred in different categories. It also contains a list of events in case Internet is down or we are otherwise unable to send data to the server. If you want to send data in bulks instead of continuously, this is the place to implement it. Useful function for other parts of the program is write\*() function, togglePaused(), updateSID(), retPaused(); and toggleServerStatic(). This is the API that this class provides. The write\*() functions should handle how the data is transmitted to the server.

1. Author Robin Stenvi - 2012-01-21

Robin Stenvi - 2012-05-04 (Last modified)

### 2.17.2 Constructor & Destructor Documentation

#### 2.17.2.1 handleData::handleData ( )

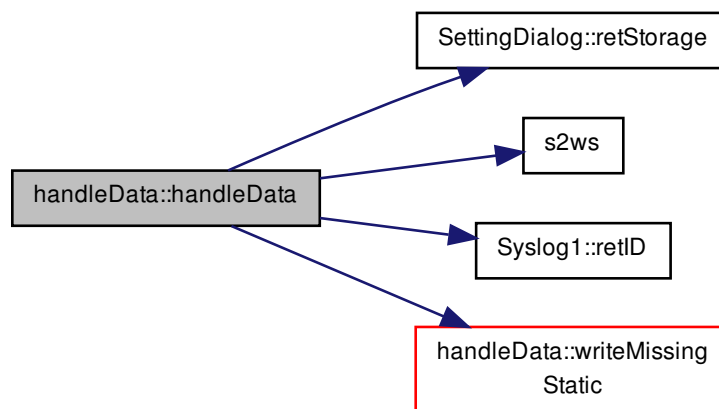
Initiate all variables and create transmission object.

1. Author Robin Stenvi

#### Remarks

This is not called between start and stop, so be vary of what is initialized here.

Here is the call graph for this function:



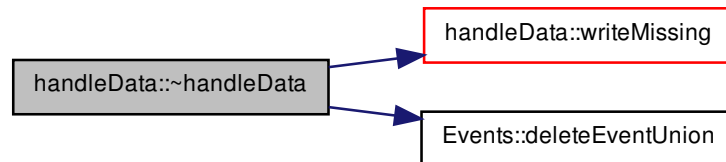
#### 2.17.2.2 handleData::~~handleData ( )

Need to delete everything it has used and flush all remaining output, if any.

Here we stop any threads that are still running and if there are events in memory we have not sent yet, we will write them to local file and send them on next startup.

1. Author Robin Stenvi

Here is the call graph for this function:



### 2.17.3 Member Function Documentation

#### 2.17.3.1 `char * handleData::getTimestamp ( const char * from )` [protected]

Retrives timestamp from syslog string in a more readable format.

1. Author Robin Stenvi

##### Parameters

<code>in</code>	<code>from</code>	The complete syslog string that is sent to the server.
-----------------	-------------------	--

##### Returns

Returns a string with the time and date only

#### 2.17.3.2 `std::string handleData::getTimestamp ( std::string from )` [protected]

Retreives the timestamp from a syslog event string.

1. Author Robin Stenvi

##### Parameters

<code>in</code>	<code>from</code>	The entire syslog event string
-----------------	-------------------	--------------------------------

**Returns**

Returns the timestamp.

Here is the call graph for this function:



### 2.17.3.3 `std::string handleData::getTimestamp ( bool date = false )` [protected]

Retrieves a timestamp with seconds accuracy.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>date</i>	Specify whether you should have date with time or just time.
-----------	-------------	--

**Returns**

Returns an `std::string` with the format (YYYY-MM-DD) HH:MM:SS. If an error happened, the time will be `ErrorClock` or `ErrorDate`.

### 2.17.3.4 `std::string handleData::getCsvMouse ( MouseInfo input, int ev, int rel = 0 )` [protected]

Retrives a mouse event in CSV format, can be used if you need to write local storage.

1. Author Robin Stenvi

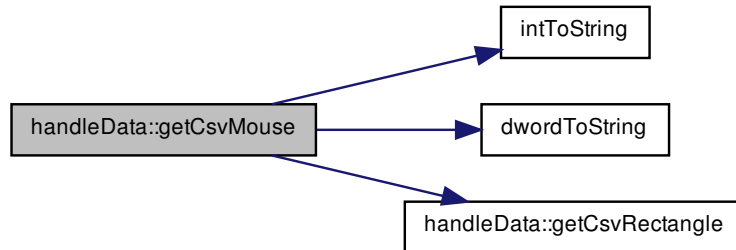
**Parameters**

<i>in</i>	<i>input</i>	Structure containing information about a mouse event
<i>in</i>	<i>ev</i>	The correct event number
<i>in</i>	<i>rel</i>	A pointer to which this event is in relation to

**Returns**

Returns a mouse event in CSV format.

Here is the call graph for this function:



### 2.17.3.5 `std::string handleData::getCsvUIA ( eventInfoUnion input, int ev, int rel = 0 )` [protected]

Retrieves a software event in CSV format, can be used if you need to write local storage.

1. Author Robin Stenvi

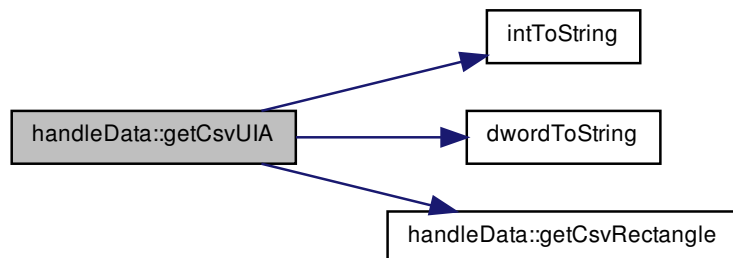
**Parameters**

in	<i>input</i>	Structure containing information about a software event
in	<i>ev</i>	The correct event number
in	<i>rel</i>	A pointer to which this event is in relation to

**Returns**

Returns a software event in CSV format.

Here is the call graph for this function:





### 2.17.3.6 `std::string handleData::getCsvKey ( KeyInfo input, int ev, int rel = 0 )` [protected]

Retrives a key event in CSV format, can be used if you need to write local storage.

1. Author Robin Stenvi

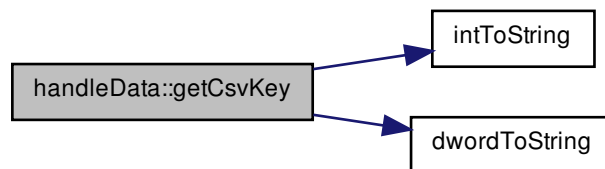
#### Parameters

in	<i>input</i>	Structure containing information about a key event
in	<i>ev</i>	The correct event number.
in	<i>rel</i>	A pointer to which this event is in relation to

#### Returns

Returns a key event in CSV format.

Here is the call graph for this function:



### 2.17.3.7 `std::string handleData::getCsvHW ( sysResources * HW )` [protected]

Retrives a system resource event in CSV format, can be used if you need to write local storage.

1. Author Robin Stenvi

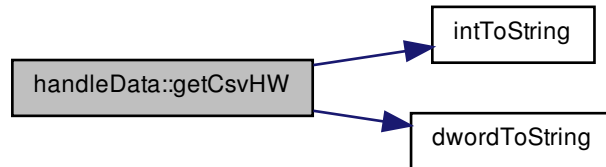
#### Parameters

in	<i>HW</i>	Structure containing information about a system resource event
----	-----------	--

**Returns**

Returns a system resource event in CSV format. Returns ErrorCsv if we fail.

Here is the call graph for this function:



### 2.17.3.8 `std::string handleData::getCsvDev ( deviceInfo dev )` [protected]

Retrieves a device insert/remove event in CSV format, can be used if you need to write local storage.

1. Author Robin Stenvi

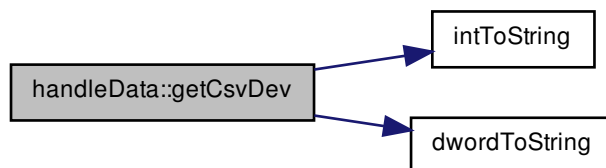
**Parameters**

<code>in</code>	<code>dev</code>	Structure containing information about a device insert/remove event
-----------------	------------------	---

**Returns**

Returns a system resource event in CSV format.

Here is the call graph for this function:



### 2.17.3.9 `std::string handleData::getCsvScreen ( Screen screen )` [protected]

Retrieves CSV format of a physical screen change event.

1. Author Robin Stenvi

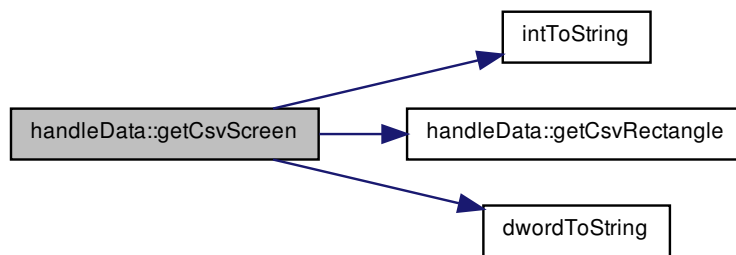
#### Parameters

<i>in</i>	<i>screen</i>	A structure with information about the screen
-----------	---------------	---

#### Returns

Returns the string in CSV format or ErrorCsv

Here is the call graph for this function:



#### 2.17.3.10 `std::string handleData::getCsvHID ( HIDDevice device )` [protected]

Retrieves CSV version of a HID change.

1. Author Robin Stenvi

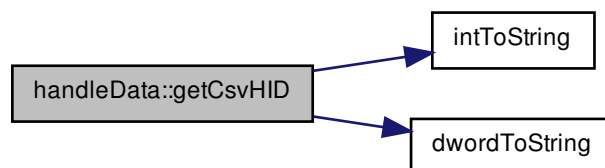
#### Parameters

<i>in</i>	<i>device</i>	Struct containing information about the change.
-----------	---------------	---

#### Returns

Returns the string in CSV format or ErrorCsv

Here is the call graph for this function:



**2.17.3.11** `std::string handleData::getCsvKeyboard ( KeyboardDevice kd )` [protected]

Retrieves CSV format with informatio about a physical keyboard.

1. Author Robin Stenvi

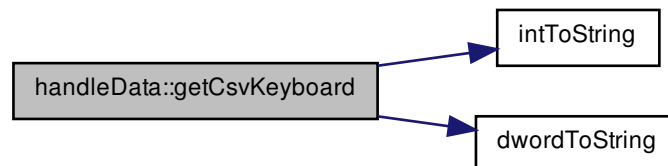
**Parameters**

<i>in</i>	<i>kd</i>	Structure containing information about the keyboard
-----------	-----------	---

**Returns**

Returns the string in CSV format or ErrorCsv

Here is the call graph for this function:

**2.17.3.12** `std::string handleData::getCsvBelt ( int evType, DWORD time )` [protected]

Retreives a string in CSV format for the start/stop/pause/resume messages.

1. Author Robin Stenvi The line has the following format `event,start/stop/pause/resume,time`

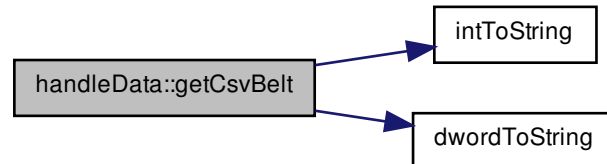
**Parameters**

<i>in</i>	<i>evType</i>	Says whether it is start/stop/pause/resume, can be one of the following values: LISTENER_START, LISTENER_STOP, LISTENER_PAUSE, LISTENER_RESUME, if not it returns ERROR
<i>in</i>	<i>time</i>	Timestamp for when the event happened.

**Returns**

Returns a string representing the whole event in CSV format, ends in newline.

Here is the call graph for this function:



### 2.17.3.13 `std::string handleData::getCsvRectangle ( RECT * rect )` [protected]

Retrieves a rectangle in CSV format, can be used if you need to write local storage.

1. Author Robin Stenvi

**Parameters**

<code>in</code>	<code><i>rect</i></code>	The pointer to a rectangle
-----------------	--------------------------	----------------------------

**Returns**

Returns a rectangle in CSV format: bottom,right,top,left

### 2.17.3.14 `std::string handleData::getFormatMouse ( MouseInfo input, int rel = 0 )` [protected]

Retrieves the complete structured data that should be sent in any mouse event.

1. Author Robin Stenvi

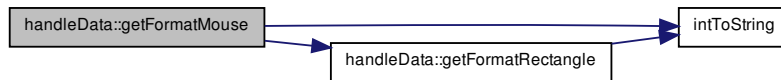
**Parameters**

<code>in</code>	<code><i>input</i></code>	A single mouse event
<code>in</code>	<code><i>rel</i></code>	Which event is this correlated with

**Returns**

Structured data to be used as part of a message to the server

Here is the call graph for this function:



### 2.17.3.15 `std::string handleData::getFormatUIA ( eventInfoUnion input, int rel = 0 )` [protected]

Gets the structured data of a software event.

1. Author Robin Stenvi

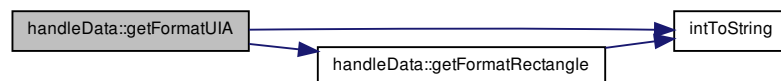
**Parameters**

<i>in</i>	<i>input</i>	A software event
<i>in</i>	<i>rel</i>	Which event is this correlated with

**Returns**

Structured data to be used as part of a message to the server

Here is the call graph for this function:



### 2.17.3.16 `std::string handleData::getFormatKey ( KeyInfo input, int rel = 0 )` [protected]

Gets the structured data of a key event.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>input</i>	A key event
<i>in</i>	<i>rel</i>	Which event is this correlated with

**Returns**

Structured data to be used as part of a message to the server

Here is the call graph for this function:



### 2.17.3.17 `std::string handleData::getFormatHW ( sysResources * HW )` [protected]

Gets the structured data of a HW average event.

1. Author Robin Stenvi

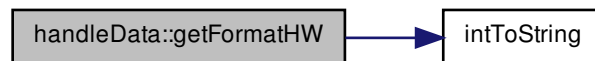
#### Parameters

<code>in</code>	<i>HW</i>	A HW average event
-----------------	-----------	--------------------

#### Returns

Structured data to be used as part of a message to the server

Here is the call graph for this function:



### 2.17.3.18 `std::string handleData::getFormatKeyboard ( KeyboardDevice kd )` [protected]

Retrieves the structured data with informatio about a physical keyboard.

1. Author Robin Stenvi

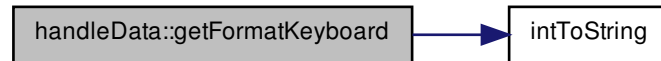
#### Parameters

<code>in</code>	<i>kd</i>	Structure containing information about the keyboard
-----------------	-----------	---

**Returns**

Returns the string in CSV format or ErrorServer

Here is the call graph for this function:



### 2.17.3.19 `std::string handleData::getFormatScreen ( Screen screen )` [protected]

Gets the structured data of a screen event.

1. Author Robin Stenvi

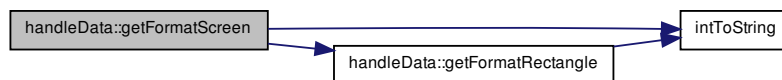
**Parameters**

<code>in</code>	<code>screen</code>	A screen event
-----------------	---------------------	----------------

**Returns**

Structured data to be used as part of a message to the server

Here is the call graph for this function:



### 2.17.3.20 `std::string handleData::getFormatHID ( HIDDevice device )` [protected]

Gets the structured data of a input device event.

1. Author Robin Stenvi

**Parameters**

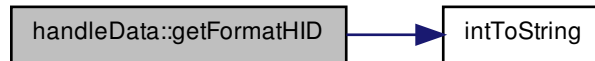
<code>in</code>	<code>device</code>	A input device event
-----------------	---------------------	----------------------



**Returns**

Structured data to be used as part of a message to the server

Here is the call graph for this function:



### 2.17.3.21 `std::string handleData::getFormatRectangle ( RECT * rect )` [protected]

Returns rectangle in the form of structured data that can be sent to the server.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>rect</i>	The rectangle which we are going to make a string out of.
-----------	-------------	---

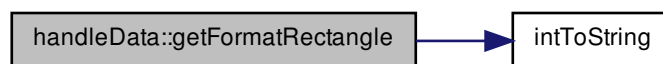
**Returns**

Returns the actual string in the following format: bottY="a" topY="b" leftX="c" rightX="d", If anythin fails, ErrorFormatRectangle is returned.

**Remarks**

Only formats the rectangle part, this is one part of a larger message

Here is the call graph for this function:



### 2.17.3.22 `std::string handleData::getEventToServer ( eventInfoUnion * out, int backRef, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to a software event.

1. Author Robin Stenvi

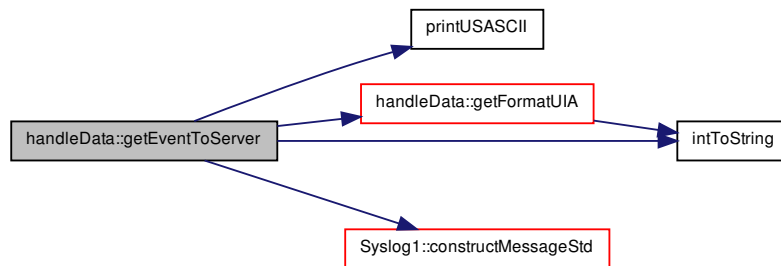
#### Parameters

<i>in</i>	<i>out</i>	The structure representing an event
<i>in</i>	<i>backRef</i>	An int representing which message this correlates with
<i>in</i>	<i>ev</i>	Which event number this is, default is zero, which means the current event number

#### Returns

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



#### 2.17.3.23 `std::string handleData::getMouseToServer ( MouseInfo out, int backRef, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to a mouse event.

1. Author Robin Stenvi

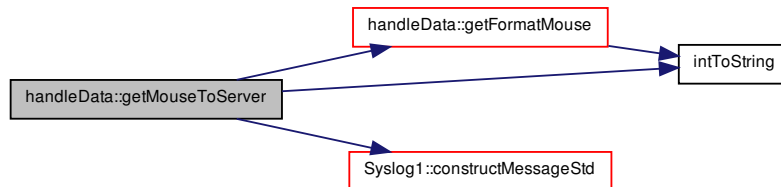
#### Parameters

<i>in</i>	<i>out</i>	The structure representing an event
<i>in</i>	<i>backRef</i>	An int representing which message this correlates with
<i>in</i>	<i>ev</i>	Which event number this is, default is zero, which means the current event number

**Returns**

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



### 2.17.3.24 `std::string handleData::getKeyToServer ( KeyInfo out, int backRef, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to a key event.

1. Author Robin Stenvi

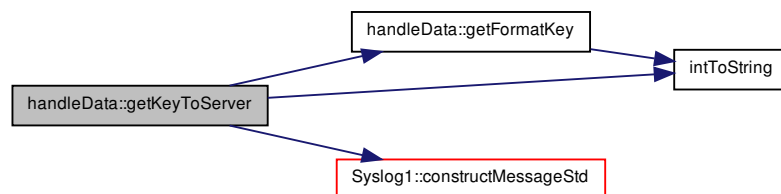
**Parameters**

<i>in</i>	<i>out</i>	The structure representing an event
<i>in</i>	<i>backRef</i>	An int representing which message this correlates with
<i>in</i>	<i>ev</i>	Which event number this is, default is zero, which means the current event number

**Returns**

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



### 2.17.3.25 `std::string handleData::getHWToServer ( sysResources * HW, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to a system resource usage.

1. Author Robin Stenvi

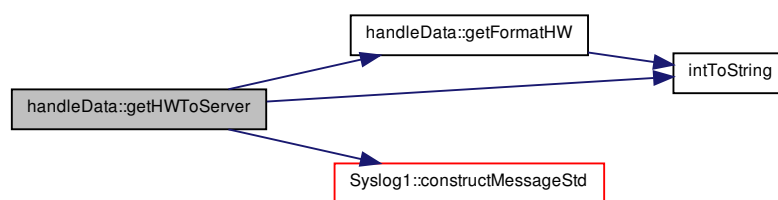
#### Parameters

in	<i>HW</i>	The structure representing an event
in	<i>ev</i>	Which event number this is, default is zero, which means the current event number

#### Returns

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



#### 2.17.3.26 std::string handleData::getDevToServer ( deviceInfo dev, int ev = 0 ) [protected]

Retrieves a syslog compatible message corresponding to a device insert /remove.

1. Author Robin Stenvi

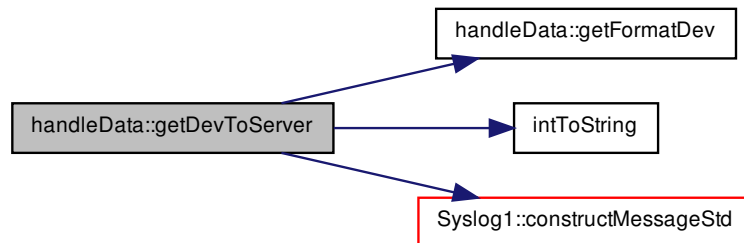
#### Parameters

in	<i>dev</i>	The structure representing an event
in	<i>ev</i>	Which event number this is, default is zero, which means the current event number

**Returns**

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



### 2.17.3.27 `std::string handleData::getScreenToServer ( Screen screen, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to a physical screen change event.

1. Author Robin Stenvi

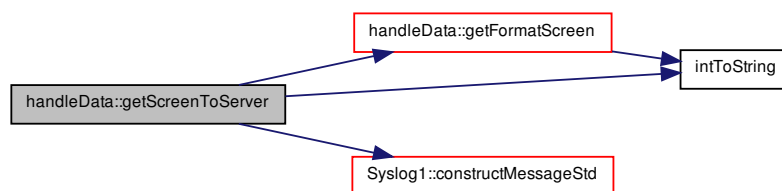
**Parameters**

in	<i>screen</i>	The structure representing an event
in	<i>ev</i>	Which event number this is, default is zero, which means the current event number

**Returns**

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



### 2.17.3.28 `std::string handleData::getHIDToServer ( HIDDevice device, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to a physical device.

1. Author Robin Stenvi

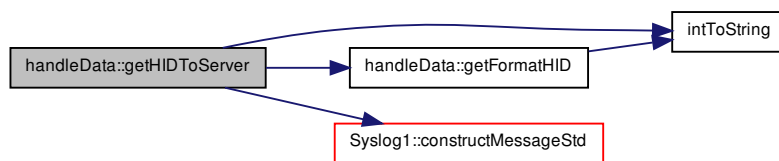
#### Parameters

in	<i>device</i>	The structure representing an event
in	<i>ev</i>	Which event number this is, default is zero, which means the current event number

#### Returns

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



#### 2.17.3.29 `std::string handleData::getKeyboardToServer ( KeyboardDevice kd, int ev = 0 )` [protected]

Retrieves a syslog compatible message corresponding to information about a keyboard.

1. Author Robin Stenvi

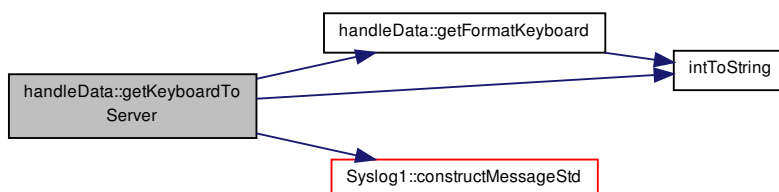
#### Parameters

in	<i>kd</i>	The structure representing an event
in	<i>ev</i>	Which event number this is, default is zero, which means the current event number

#### Returns

Returns the message on success or a syslog compatible error message on failure.

Here is the call graph for this function:



### 2.17.3.30 `std::string handleData::getBeltToServer ( int evType, DWORD time )` [protected]

Get a syslog message corresponding to a start/stop/pause/resume in belt event.

1. Author Robin Stenvi

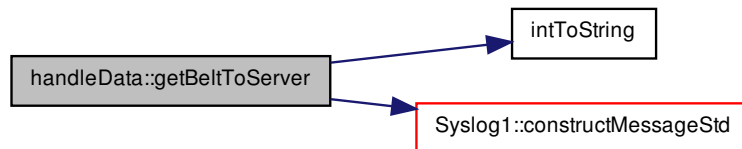
#### Parameters

in	<i>evType</i>	Value between 1 and 4 determining if we start/stop...
in	<i>time</i>	The tick count when the event happened

#### Returns

Returns a syslog compatible string.

Here is the call graph for this function:



### 2.17.3.31 `std::string handleData::getDescSentenceMouse ( MouseInfo input, std::string finalTime )` [protected]

Gets a descriptive sentence for mouse events that can be displayed to the user.

1. Author Robin Stenvi

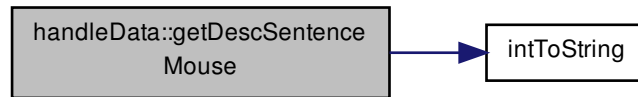
#### Parameters

in	<i>input</i>	A mouse event
in	<i>finalTime</i>	The time the event occurred

**Returns**

A string that can be printed to the user

Here is the call graph for this function:



### 2.17.3.32 `std::string handleData::getDescSentenceKey ( KeyInfo input, std::string finalTime )` [protected]

Gets a descriptive sentence for key events that can be displayed to the user.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>input</i>	A key event
<i>in</i>	<i>finalTime</i>	The time the event occurred

**Returns**

{A string that can be printed to the user}

### 2.17.3.33 `void handleData::sendToServer ( std::string str )` [protected]

Send a single event to the server, the string should already be formatted correctly.

If it is unable to send the event, it will store it temporarily and send it later. If too many events happen and we are not able to send it to the server, we will delete them, but keep the original event number so the server know how many events are missing.

1. Author Robin Stenvi

**Parameters**

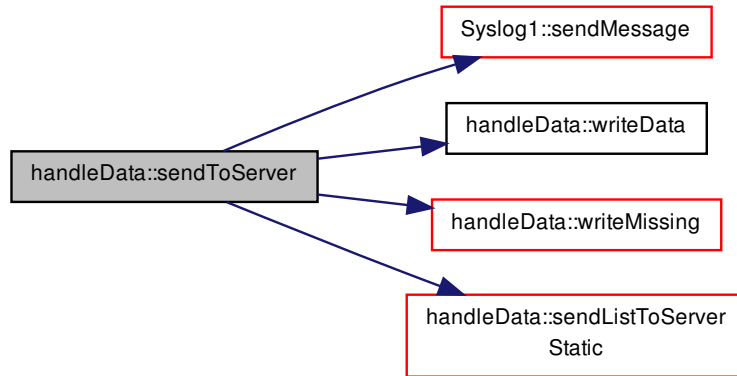
<i>in</i>	<i>str</i>	points to a syslog compatible message that can be sent directly to the server
-----------	------------	---



**Remarks**

Might do local storage to file here, so it can be sent later, the timestamp will contain a date so there is no problem doing that.

Here is the call graph for this function:



### 2.17.3.34 void handleData::sendListToServer ( ) [protected]

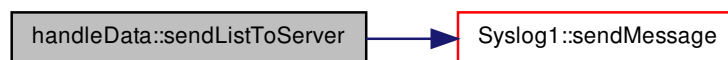
Send all that is stored temporarily and send it to the server.

1. Author Robin Stenvi

**Remarks**

Should run this on a separate thread to avoid that the program become unresponsive. Does not check if messages were successfully sent. This function will clear `sendList`

Here is the call graph for this function:



### 2.17.3.35 void handleData::sendData ( std::string data, UINT type = 0 ) [protected]

Used to send string back to the GUI, it makes the decision whether it should be displayed or not.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>data</i>	The string that should be sent.
<i>in</i>	<i>type</i>	0 means write it to file, non-zero means display it to the GUI, type then says what type of event it was.

#### 2.17.336 BOOL handleData::writeData ( std::string *data* ) [protected]

Writes whatever data that comes in to file.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>data</i>	The string that should be written.
-----------	-------------	------------------------------------

#### Returns

TRUE if we can write to file, FALSE otherwise.

#### 2.17.337 void handleData::sendFullListToServer ( ) [protected]

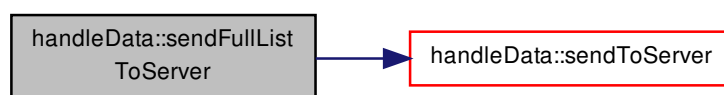
Sends everything we have stored temporarily in a list to the server.

1. Author Robin Stenvi

#### Remarks

This function can only run on one thread, and that thread need should only do this. It will read and send 10 events, then it will lock the mutex and delete those 10 events. Since this function is always working on the first events and other function add to the end, we only need to lock the mutex when deleting events. If the events happen rapidly enough, this thread will never end.

Here is the call graph for this function:

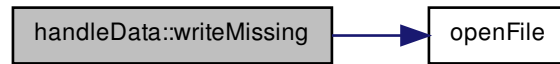


#### 2.17.338 void handleData::writeMissing ( ) [protected]

Should write any files that we didn't have time to send, for whatever reason, should also mark it in the profile file.

1. Author Robin Stenvi

Here is the call graph for this function:



### 2.17.3.39 void handleData::sendMissing ( ) [protected]

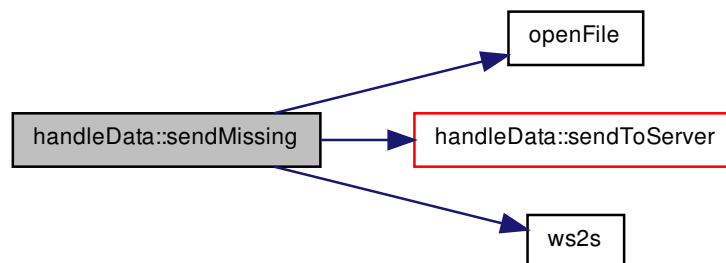
Should send any files that we were unable to send the last time we ran the program.

1. Author Robin Stenvi

#### Remarks

Unknown how much time this will take, so it is probably best to call it on a separate thread.

Here is the call graph for this function:



### 2.17.3.40 void handleData::writeAll ( std::string str ) [protected]

Writes an event to file or server.

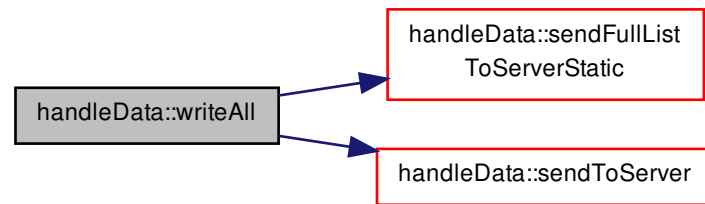
Whether it writes it to server or file depends on what the current setting is. Whenever you need to send an event, you can call this function and it will handle the remaining logic, but the string must be formatted correctly.

1. Author Robin Stenvi

#### Parameters

in	<i>str</i>	The string that should be logged.
----	------------	-----------------------------------

Here is the call graph for this function:



#### 2.17.3.41 void handleData::writeMissingStatic ( void \* *p* ) [static], [protected]

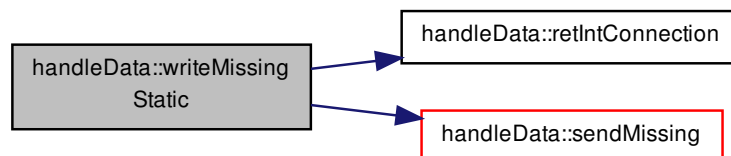
Wrapper to `sendMissing()`, used when calling on a thread, will check if Internet is available every 5 seconds and start sending as soon as Internet is available.

1. Author Robin Stenvi

##### Parameters

<i>in</i>	<i>p</i>	Unused
-----------	----------	--------

Here is the call graph for this function:



#### 2.17.3.42 bool handleData::startNewSession ( std::string *bufTmp* ) [protected]

Initializes all the variables needed to start a session.

1. Author Robin Stenvi

##### Parameters

<i>in</i>	<i>bufTmp</i>	A string containing the current date and time, with the format Y-m-d H-M-S
-----------	---------------	--

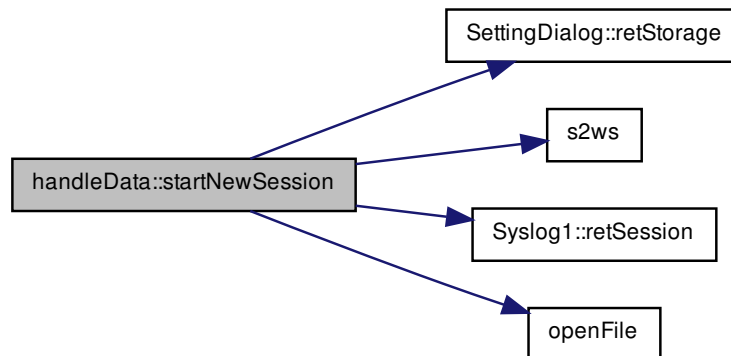
**Remarks**

This function is not enough to start a new session, to start or stop a new session, you should use writeTime with the appropriate parameter.

**Returns**

Returns true on success, false on failure.

Here is the call graph for this function:

**2.17.3.43 void handleData::stopCurrentSession ( ) [protected]**

Closes a running session.

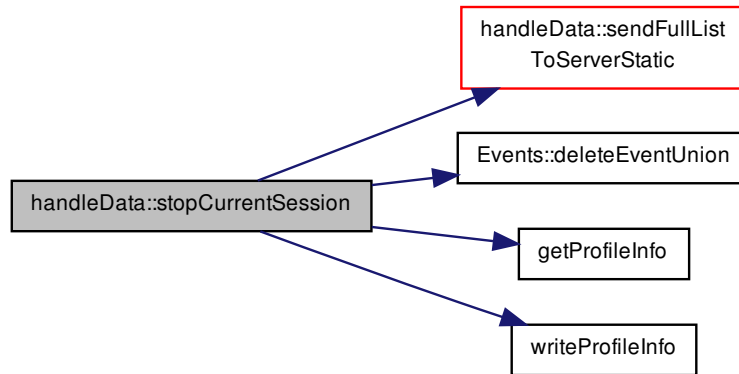
1. Author Robin Stenvi This function will flush and close any files and start a thread that sends remaining events to server. This func-

tion will also delete any variables we don't need anymore.

## Remarks

This function should always be called before exiting the application. The user will not probably do it, so we have to do it.

Here is the call graph for this function:



#### 2.17.3.44 void handleData::accessKey ( KeyInfo out ) [protected]

If the access key for pause (`|alt|+|pause|`) has been hit we tell the GUI that logging should be paused.

1. Author Robin Stenvi

## Parameters

<i>in</i>	<i>out</i>	The key event that occurred.
-----------	------------	------------------------------

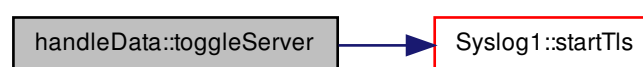
#### 2.17.3.45 void handleData::toggleServer ( ) [protected]

Start or stop the connection to the server, depending on what the current status is.

Do not call this directly, instead call `handleData::toggleServerStatic`. If the last time we called this function it failed, we will wait 10 seconds. That way, this function can be called repeatedly when we fail.

1. Author Robin Stenvi

Here is the call graph for this function:



### 2.17.3.46 `char * handleData::writeTime ( const char * sentence, int eventType )`

Write time and a message to server, used at pause/resume/stop/start.

1. Author Robin Stenvi

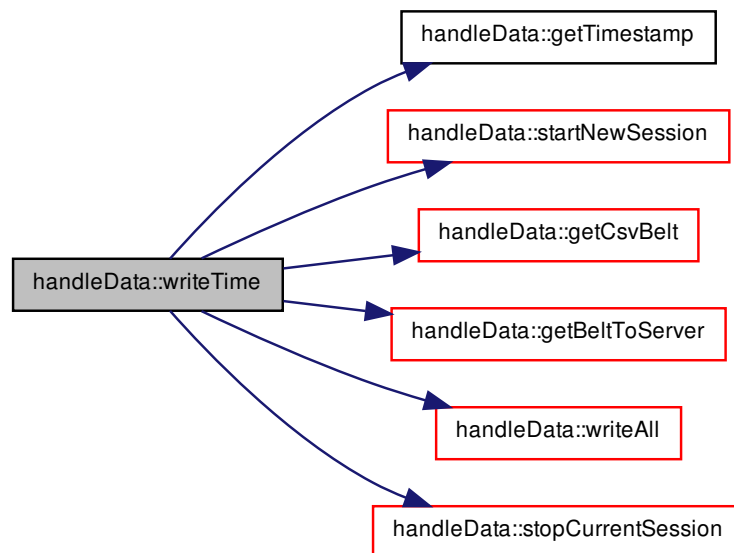
#### Parameters

in	<i>sentence</i>	The string that should be written to the user, does not affect what is sent to the server
in	<i>eventType</i>	Says whether the event is sart/stop/pause/resume

#### Returns

A string containing the sentence that is sent in and a timestamp, saying when the event occurred.

Here is the call graph for this function:



### 2.17.3.47 `bool handleData::writeEventToServer ( eventInfoUnion * out )`

Takes an UI AUtimation event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

#### Parameters

in	<i>out</i>	Pointer to the software event, should be allocated with new
----	------------	---

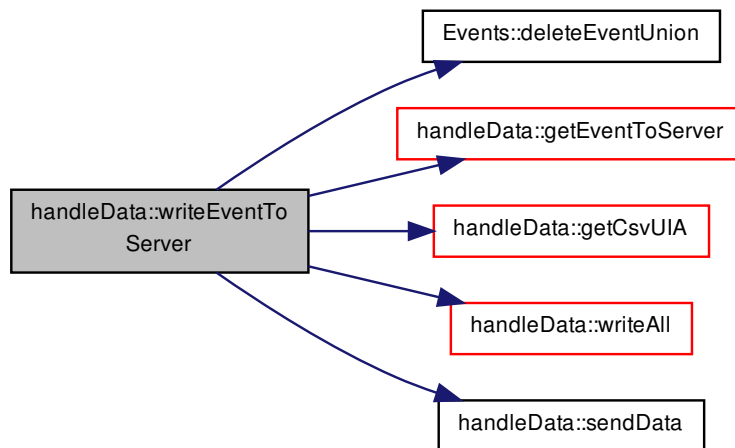
**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources

**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:

**2.17.3.48 bool handleData::writeMouseToServer ( MouseInfo out )**

Takes a mouse event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>out</i>	Mouse event
-----------	------------	-------------

**Returns**

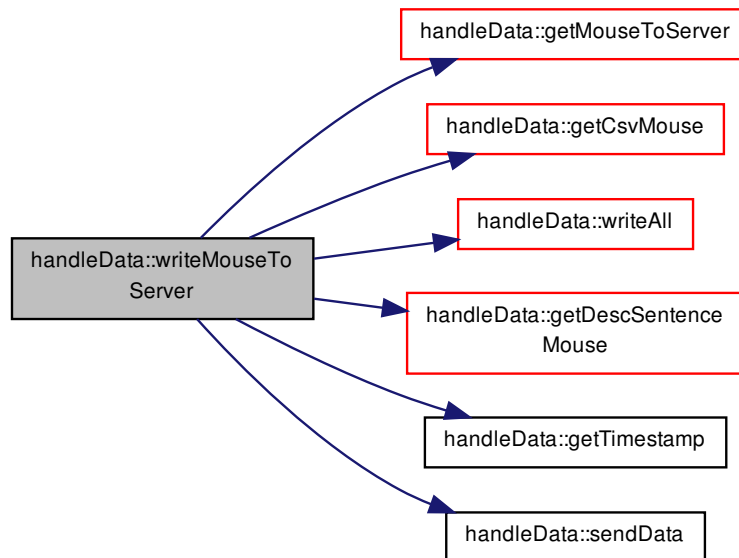
Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources



**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:



#### 2.17.3.49 bool handleData::writeKeyToServer ( KeyInfo out )

Takes a key event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>out</i>	Key event
-----------	------------	-----------

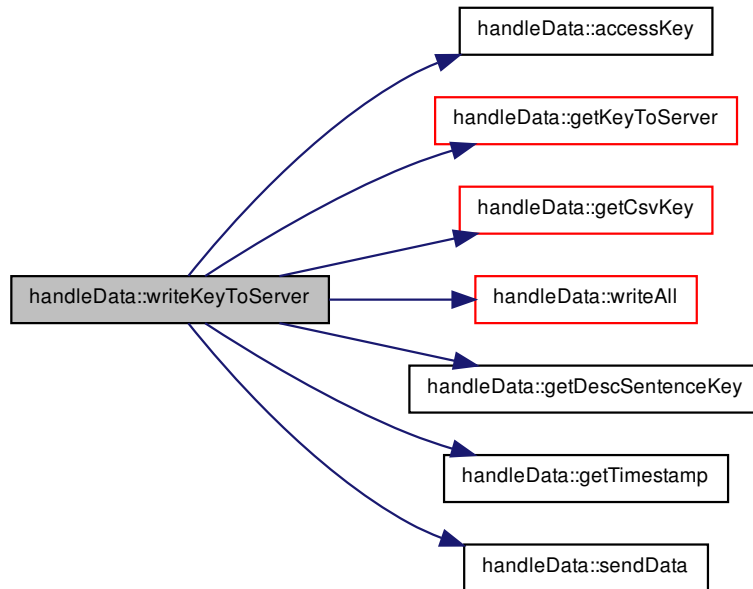
**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources

**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:



### 2.17.3.50 `bool handleData::writeHWToServer ( sysResources * HW )`

Takes a hardware event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

**Parameters**

<code>in</code>	<i>HW</i>	Pointer to the hardware event
-----------------	-----------	-------------------------------

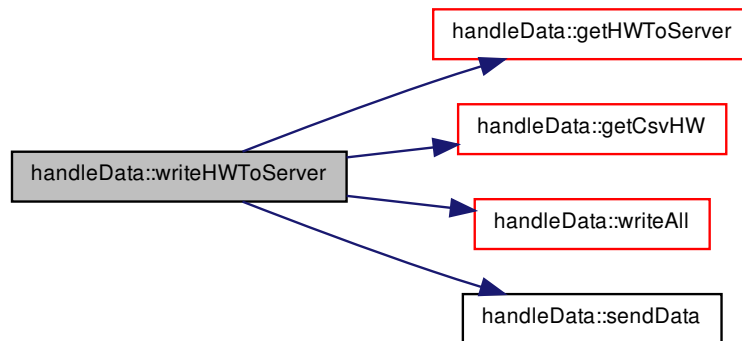
**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources

**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:

**2.17.3.51 bool handleData::writeDevToServer ( deviceInfo dev )**

Takes a device event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

**Parameters**

<code>in</code>	<code>dev</code>	Device event
-----------------	------------------	--------------

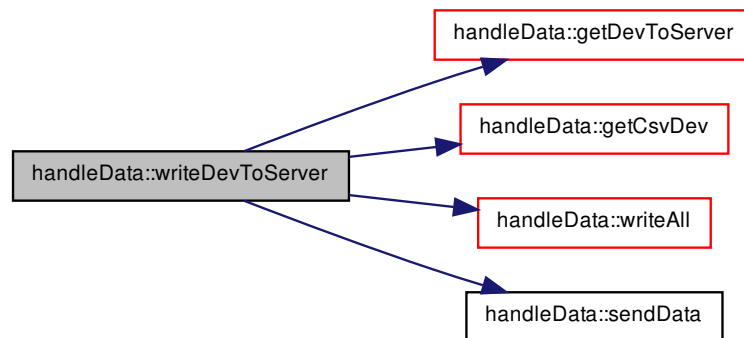
**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources

**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:

**2.17.3.52 bool handleData::writeScreenToServer ( Screen *screen* )**

Takes a screen event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>screen</i>	Screen event
-----------	---------------	--------------

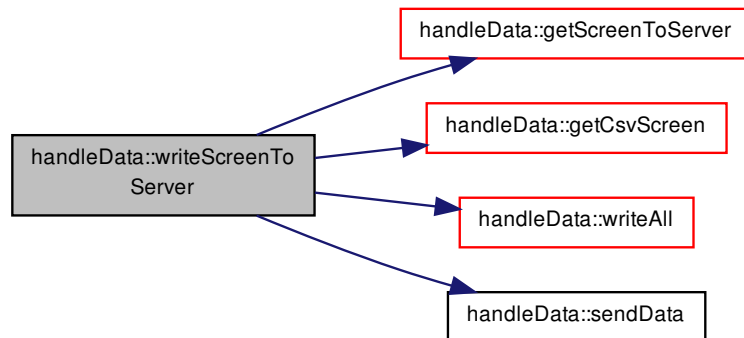
**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources

**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:



### 2.17.3.53 `bool handleData::writeHIDToServer ( HIDDevice device )`

Takes an input device event and writes it to the server, GUI and file if necessary.

1. Author Robin Stenvi

**Parameters**

<code>in</code>	<code>device</code>	Input device event
-----------------	---------------------	--------------------

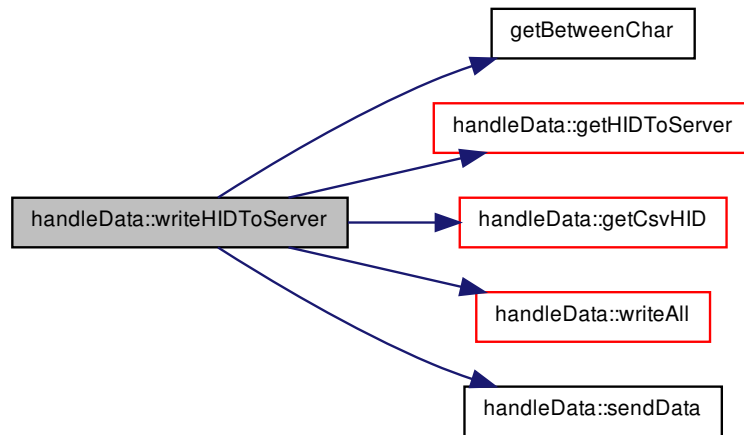
**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources

**Remarks**

Need to make sure the above is correct

Here is the call graph for this function:



#### 2.17.3.54 `bool handleData::writeKeyboardToServer ( KeyboardDevice kd )`

Writes information about the keyboard to the server.

1. Author Robin Stenvi

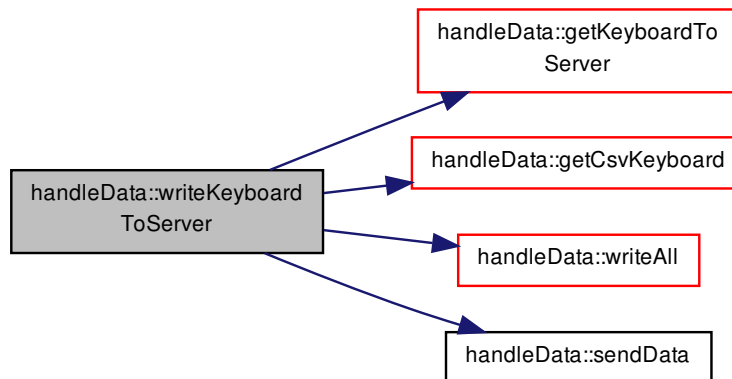
**Parameters**

<i>in</i>	<i>kd</i>	Information about the keyboard
-----------	-----------	--------------------------------

**Returns**

Returns false if we filtered out the event or it was otherwise not sent, true if we sent the event to all configured sources.

Here is the call graph for this function:



#### 2.17.3.55 `bool handleData::writeStringToServer ( std::string str )`

Takes whatever string it gets and write it to the server.

1. Author Robin Stenvi

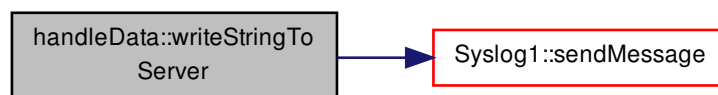
##### Parameters

<code>in</code>	<code>str</code>	The string that should be written
-----------------	------------------	-----------------------------------

##### Returns

Returns true if we are successful, false otherwise.

Here is the call graph for this function:



#### 2.17.3.56 `void handleData::toggleServerStatic ( void * p ) [static]`

Static helper function, so we can call `toggleServer()`.

1. Author Robin Stenvi

#### Parameters

in	<i>p</i>	An object to handleData
----	----------	-------------------------

#### 2.17.3.57 void handleData::sendListToServerStatic ( void \* *p* ) [static]

Static function you can use when starting a new thread, function sends the list we kept when server connection was unable.

1. Author Robin Stenvi

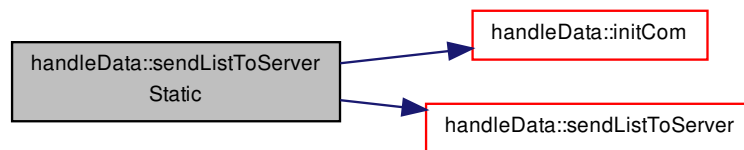
#### Parameters

in	<i>p</i>	Unused
----	----------	--------

#### Remarks

This function initialized the COM library.

Here is the call graph for this function:



#### 2.17.3.58 void handleData::sendFullListToServerStatic ( void \* *p* ) [static]

Sends completeList to server, static functions that can be used when starting a new thread.

1. Author Robin Stenvi

#### Parameters

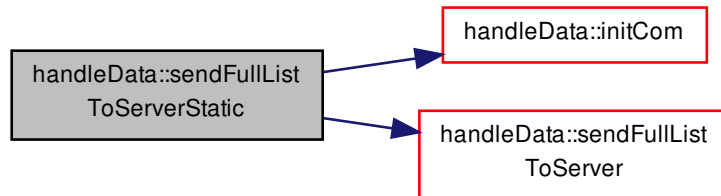
in	<i>p</i>	Unused
----	----------	--------



**Remarks**

This function initialized the COM library.

Here is the call graph for this function:

**2.17.3.59 bool handleData::retPaused ( )**

Returns the current value of the pause variable.

1. Author Robin Stenvi

**Returns**

true if paused otherwise falsed, also if it has stopped

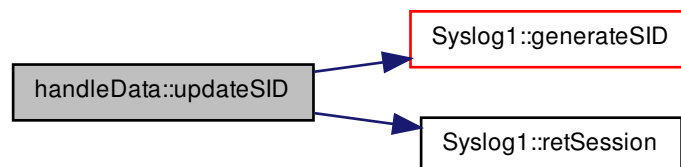
**2.17.3.60 void handleData::updateSID ( )**

Update and set the session number.

Updates and sets the session number by calling the syslogs generate SID. If no SID is set it is set to 0. If the SID is set is is incremented and the handleDatass SID is set to the new SID.

1. Author Magnus Øverbø - 12.03.2013

Here is the call graph for this function:

**2.17.3.61 void handleData::setLogPort ( int port )**

Changes the port number we should use.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>port</i>	The port number that should be used
-----------	-------------	-------------------------------------

Here is the call graph for this function:



#### 2.17.3.62 void handleData::setLogAddr ( std::wstring *addr* )

Changes the server adress that we should use.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>addr</i>	The server address
-----------	-------------	--------------------

Here is the call graph for this function:



#### 2.17.3.63 bool handleData::initCom ( )

Initiate the COM library, useful if you start a new thread on a static function.

1. Author Robin Stenvi

**Returns**

Returns true if we are successful, false if we don't succeed.

Here is the call graph for this function:

**2.17.4 Member Data Documentation****2.17.4.1** `std::vector<std::string> handleData::serverList` `[private]`

All the events we have not sent to the server yet, they are held here temporarily because the connection to the server is down for whatever reason.

When we reach MAX\_STORAGE events we clear it regardless if they are sent or not.

**2.17.4.2** `const int handleData::MAX_STORAGE = 10000` `[static], [private]`

Number of elements we can store locally before sending to server, each event is about 100B.

Is used if connection is lost, now we store 1MB before removing it all if we still have not gotten a connection

**2.18 HIDDevice Struct Reference**

Information about an input device.

**Public Attributes**

- `std::string name`  
*The full name of the device.*
- `int type`  
*mouse/keyboard = 0/1, directly from RAWINPUTHEADER*
- `HANDLE ID`  
*Internal identifier for the device.*
- `DWORD time`  
*The moment we registered it.*
- `int num`  
*This number in the list.*
- `bool first`  
*If this is the first time we have seen this.*

### 2.18.1 Detailed Description

Information about an input device.

Contains HANDLE which is used internally to identify which device we are looking at. A name which is sent to the server and can be used there to identify which input device is used. An int value which determines whether it is a mouse or a keyboard, and a timestamp, which says when we collected the information.

## 2.19 HWMonitor Class Reference

Class for monitoring Hardware usage.

### Public Member Functions

- [HWMonitor \(\)](#)  
*Only need to mark that we haven't run the CPU measurement before.*
- [~HWMonitor \(\)](#)  
*Empty destructor.*
- [bool getCpuLoad \(double &cpuLoad\)](#)  
*Gets the average CPU load between two measurements, the first measurement will not give a value.*
- [bool closeCpuLoad \(\)](#)  
*Close the CPU load measurements.*
- [bool getMemLoad \(long &memLoad\)](#)  
*Get current memory usage in %.*

### Private Attributes

- [PDH\\_STATUS statusCpu](#)  
*Variable for our query.*
- [PDH\\_FMT\\_COUNTERVALUE valueCpu](#)  
*Return value from our query.*
- [HQUERY queryCpu](#)  
*The actual query.*
- [HCOUNTER counterCpu](#)  
*The counter for each query.*
- [DWORD retCpu](#)  
*The counter type.*
- [bool firstCpu](#)  
*If this is the first time we monitor the CPU.*

### 2.19.1 Detailed Description

Class for monitoring Hardware usage.

Can monitor average CPU and memory usage, has to be called in intervals. The intervals can be irregular but there should be at least 1 second between each call. The memory usage monitor is not an average, but it doesn't change often.

1. Author Robin Stenvi

## 2.19.2 Constructor &amp; Destructor Documentation

## 2.19.2.1 HWMonitor::HWMonitor ( )

Only need to mark that we haven't run the CPU measurement before.

1. Author Robin Stenvi

## 2.19.2.2 HWMonitor::~~HWMonitor ( )

Empty destructor.

1. Author Robin Stenvi

## 2.19.3 Member Function Documentation

2.19.3.1 bool HWMonitor::getCpuLoad ( double & *cpuLoad* )

Gets the average CPU load between two measurements, the first measurement will not give a value.

1. Author Robin Stenvi

## Parameters

out	<i>cpuLoad</i>	The average CPU laod since last measurement
-----	----------------	---

## Returns

false if we were unable to get a real value for whatever reason.

## 2.19.3.2 bool HWMonitor::closeCpuLoad ( )

Close the CPU load measurements.

1. Author Robin Stenvi

## Returns

Returns true if we are successful.

2.19.3.3 bool HWMonitor::getMemLoad ( long & *memLoad* )

Get current memory usage in %.

1. Author Robin Stenvi

## Parameters

out	<i>memLoad</i>	Memory laod in percent
-----	----------------	------------------------

## Returns

Returns true if we are successful.

## 2.20 KeyboardDevice Struct Reference

Contains information about a keyboard device.

### Public Attributes

- int [keyType](#)  
*Key type (qwerty etc.) can 0 to 7.*
- int [lang](#)  
*Language identifier, contains only the lower word.*
- DWORD [time](#)  
*The moment we registered it.*
- DWORD [funcKeys](#)  
*The number of function keys.*
- DWORD [keys](#)  
*The total number of keys.*

### 2.20.1 Detailed Description

Contains information about a keyboard device.

### 2.20.2 Member Data Documentation

#### 2.20.2.1 int KeyboardDevice::keyType

Key type (qwerty etc.) can 0 to 7.

See <http://msdn.microsoft.com/en-us/library/windows/desktop/ms724336%28v=vs.85%29.aspx>

#### 2.20.2.2 int KeyboardDevice::lang

Language identifier, contains only the lower word.

See <http://msdn.microsoft.com/en-us/library/windows/desktop/dd318691%28v=vs.85%29.aspx>

## 2.21 KeyInfo Struct Reference

All variable info we need to write about a key event.

### Public Attributes

- UINT [type](#)  
*What type of key event.*
- DWORD [keyCode](#)  
*The virtual key code representing a key on the keyboard.*
- std::string [keyevent](#)  
*Says whether the event is up or down (U/D)*
- std::string [keydescription](#)  
*localized name of the keyCode, is ASCII if possible otherwise it is a name enclosed in |*
- UINT [timestamp](#)

- *Number of milliseconds since the system started, representing when the event happened.*
- [UINT flags](#)  
*Determines which function keys are currently held down.*
- [UINT count](#)  
*How many characters are sent.*

### 2.21.1 Detailed Description

All variable info we need to write about a key event.

## 2.22 Keylogger Class Reference

Collect and organizes keyboard events before they are written to disk.

### Public Member Functions

- [Keylogger \(\)](#)  
*Constructor initializes all data.*
- [~Keylogger \(\)](#)  
*Deletes all the data.*
- void [setLLEvent](#) (WPARAM wParam, KBDLLHOOKSTRUCT kbdll)  
*Sets keyinfo to contain all the relevant data, sends all previous data if keypress is up.*
- void [setPassword](#) (BOOL b)  
*If parameter is true, it will filter text until a false parameter is set.*
- void [registerState](#) ()  
*Registers the state of all keys on the keyboard.*

### Protected Member Functions

- [keyType retKeyEvent](#) (WPARAM wParam, [KeyInfo](#) &keyinfo)  
*Sets if keyevent is system key and up or down.*
- bool [setData](#) (DWORD vkCode, [keyType](#) type, [KeyInfo](#) &keyinfo)  
*Uses virtual key codes to fins a readable ASCII replacement.*
- bool [findKeyDown](#) (DWORD vkCode)  
*Finds if we have seen the key before and it has not been released.*
- int [getKeyCount](#) (DWORD vkCode)  
*Gets the number of key down events we saw, before this event.*
- bool [escapeKey](#) (std::string &fin, unsigned char ch)  
*Escapes a key into a more readable format.*

### Private Attributes

- int [counter](#)  
*Our current place in the countInfo array.*
- [countStruct](#) \* [countInfo](#)  
*Holds how many key down events we have recieved for a given key Can be used to print number of events sent, instead of printing each on a seperate line.*
- [keyType](#) [last](#)  
*The last key event we recieved.*

- BOOL [isPassword](#)  
*Determines if we have detected a password field.*
- UINT [currFlag](#)  
*The set of function keys currently pressed.*
- BYTE [allKeys](#) [256]  
*The current state of all keys on the keyboard.*

#### Static Private Attributes

- static const int [MAX\\_ARRAY](#) = 100  
*Max number of elements in the countInfo array.*

#### 2.22.1 Detailed Description

Collect and organizes keyboard events before they are written to disk.

This class recives key events immediately after they happen, they then organize all the data into the struct KeyInfo and send the data further to handleData.

1. Author Robin Stenvi - 2012-01-21

#### 2.22.2 Constructor & Destructor Documentation

##### 2.22.2.1 Keylogger::Keylogger ( )

Constructor initializes all data.

1. Author Robin Stenvi

##### 2.22.2.2 Keylogger::~Keylogger ( )

Deletes all the data.

1. Author Robin Stenvi

#### 2.22.3 Member Function Documentation

##### 2.22.3.1 **keyType** Keylogger::retKeyEvent ( WPARAM *wParam*, KeyInfo & *keyinfo* ) [protected]

Sets if keyevent is system key and up or down.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>wParam</i>	Indicate the type of key event
<i>out</i>	<i>keyinfo</i>	Strucure defining a key event

#### Returns

Returns UP, DOWN or UNKNOWN.



**2.22.3.2** `bool Keylogger::setData ( DWORD vkCode, keyType type, KeyInfo & keyinfo )` [protected]

Uses virtual key codes to find a readable ASCII replacement.

This can be used as a fail-safe, if we are unable to find the key with any other method. This function will also change `allKeys` if it was one of the meaningful system keys.

1. Author Robin Stenvi

**Parameters**

in	<i>vkCode</i>	Virtual key code
in	<i>type</i>	Whether the key was UP or DOWN.
out	<i>keyinfo</i>	Sets keydescription to a readable format.

**Returns**

Returns true if we managed to find a value for the key code

**2.22.3.3** `bool Keylogger::findKeyDown ( DWORD vkCode )` [protected]

Finds if we have seen the key before and it has not been released.

Should only be called on key down events. If we have seen that event before we will mark another key down event for that specific key.

1. Author Robin Stenvi

**Parameters**

in	<i>vkCode</i>	The virtual key code.
----	---------------	-----------------------

**Returns**

Returns true if we have seen that key before.

**2.22.3.4** `int Keylogger::getKeyCount ( DWORD vkCode )` [protected]

Gets the number of key down events we saw, before this event.

This function also cleans up in `countInfo`

1. Author Robin Stenvi

**Parameters**

in	<i>vkCode</i>	Virtual key code
----	---------------	------------------

**Returns**

Returns the number of key down event we saw before we saw a key up event.

**2.22.3.5** `bool Keylogger::escapeKey ( std::string & fin, unsigned char ch )` [protected]

Escapes a key into a more readable format.

The `ToAscii` function sometimes return values that are not readable, but they preserve the context of

the keypress, if they exist as an ASCII value it will be returned. This function tries to find a name for all possible values.

#### Parameters

out	<i>fin</i>	where to place the value
in	<i>ch</i>	The character you received from ToAscii()

1. Author Robin Stenvi

#### Returns

Returns true if we succeed, false if we fail, if we fail, you need a different way to find what the letter is, if we return false, nothing has been done with fin.

#### 2.22.3.6 void Keylogger::setLLEvent ( WPARAM wParam, KBDLLHOOKSTRUCT kbdll )

Sets keyinfo to contain all the relevant data, sends all previous data if keypress is up.

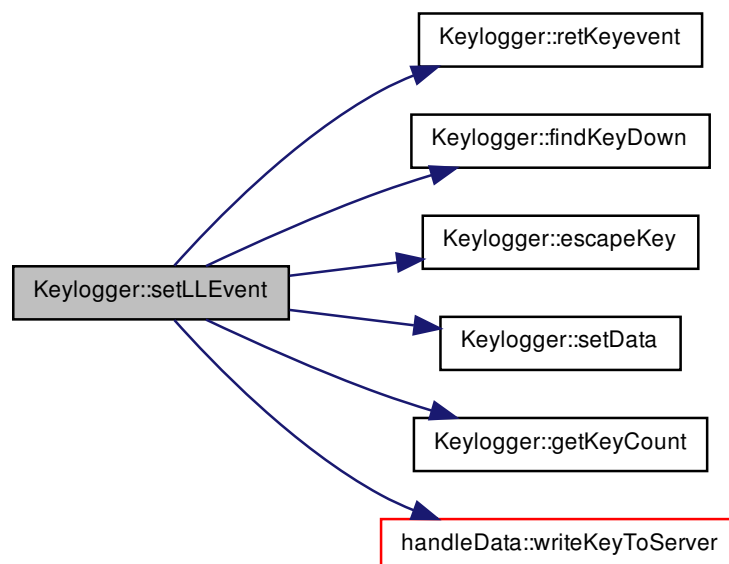
Sets all the data that we gather from a low level key event, this can be gathered for further storage and analysis or it can be written directly to file.

1. Author Robin Stenvi

#### Parameters

in	<i>wParam</i>	Type of key event
in	<i>kbdll</i>	Information about the key

Here is the call graph for this function:



**2.22.3.7 void Keylogger::setPassword ( BOOL b )**

If parameter is true, it will filter text until a false parameter is set.

1. Author Robin Stenvi

**Parameters**

in	b	If true we have detected a passowrd field.
----	---	--

**2.22.3.8 void Keylogger::registerState ( )**

Registers the state of all keys on the keyboard.

**Remarks**

This function should be called immediately before we start logging keystrokes.

1. Author Robin Stenvi

**2.23 keyType Struct Reference**

Struct to hold the number of key down we recive for some key.

**2.23.1 Detailed Description**

Struct to hold the number of key down we recive for some key.

1. Author Robin Stenvi

**2.24 handleData::lastAll Struct Reference**

Holds all the previous events, is used to find which events correlate to other events.

**Public Attributes**

- int [lastFC](#)  
*Last focus change, used to say where the keypress went.*
- int [lastUIEvent](#)  
*Last software event we saw.*
- int [lastKD](#) [MAX\_KEYS]  
*Last key down event number we saw.*
- int [lastKCount](#) [MAX\_KEYS]  
*Last key down count.*
- DWORD [lastKC](#) [MAX\_KEYS]  
*Last key code.*
- int [keyCounter](#)  
*Where in the array we are.*
- int [lastInput](#)  
*Last user input.*
- int [lastMD](#)

- *Last Mouse down.*
- int [lastMove](#)  
*Last mouse move.*

### 2.24.1 Detailed Description

Holds all the previous events, is used to find which events correlate to other events.

### 2.24.2 Member Data Documentation

#### 2.24.2.1 int handleData::lastAll::lastInput

Last user input.

Either KU, MW, MPU, MPD, or (MM?)

## 2.25 Mouse Class Reference

Collect and organizes mouse events before they are written to disk.

### Public Member Functions

- [Mouse](#) ()  
*Constructor initializes all the data.*
- [~Mouse](#) ()  
*Destructor deletes all data allocated on heap.*
- void [setLLEvent](#) (WPARAM wParam, MSLLHOOKSTRUCT msll)  
*Sets all the appropriate data in a low level event and send it to the server.*

### Protected Member Functions

- bool [printThis](#) (POINT &now)  
*returns true if the difference between to mousemoves is relevant enough to print it out*
- bool [difference](#) (double a, double b)  
*check wether the in degrees between a and b is greater than the fixed limit*
- mouseType [setmouseEvent](#) (WPARAM wParam)  
*Sets if event is a press, move or wheel.*
- bool [setLMRButton](#) (WPARAM wParam)  
*1 = left, 2 = middle, 3 = right, 4 = wheel, 0 = error*

### Private Attributes

- [MouseInfo](#) [mouseinfo](#)  
*Information about one mouse event.*
- POINT \* [lastMv](#)  
*Last mouse move we saw.*
- POINT \* [lastWrit](#)  
*Last mouse move we registered and wrote down.*
- double [oldDegrees](#)  
*Last degree change we saw, used for mouse compression.*

- double `DEGREE_CHANGE`  
*Max degree change we allow before writing down the mouse move.*

#### Static Private Attributes

- static const UINT `REAL_DISTANCE` = 10  
*Max distance we can get before registering the mouse move.*

#### 2.25.1 Detailed Description

Collect and organizes mouse events before they are written to disk.

This class recives mouse events immediately after they happen, they then organize all the data into the struct `MouseInfo` and send the data further to `handleData`.

1. Author Robin Stenvi - 2012-01-21

#### 2.25.2 Constructor & Destructor Documentation

##### 2.25.2.1 `Mouse::Mouse ( )`

Constructor initializes all the data.

1. Author Robin Stenvi

##### 2.25.2.2 `Mouse::~~Mouse ( )`

Destructor deletes all data allocated on heap.

1. Author Robin Stenvi

#### 2.25.3 Member Function Documentation

##### 2.25.3.1 `bool Mouse::printThis ( POINT & now )` [protected]

returns true if the difference between to mousemoves is relevant enough to print it out

1. Author Robin Stenvi

#### Parameters

<code>in, out</code>	<code>now</code>	The current X, Y coordinates
----------------------	------------------	------------------------------

**Returns**

Returns true if we should print this event, false if we should not print it.

Here is the call graph for this function:

**2.25.3.2 bool Mouse::difference ( double *a*, double *b* ) [protected]**

check wether the in degrees between a and b is greater than the fixed limit

1. Author Robin Stenvi

**Parameters**

in	<i>a</i>	The current number of degrees
in	<i>b</i>	The last number of degrees

**Returns**

Returns true if the difference between a and b is larger than DEGREE\_CHANGE

**2.25.3.3 mouseType Mouse::setmouseEvent ( WPARAM *wParam* ) [protected]**

Sets if event is a press, move or wheel.

1. Author Robin Stenvi

**Parameters**

in	<i>wParam</i>	Says what type of mouse event that happened, foolows from the hook.
----	---------------	---

**Returns**

Returns PRESS, WHEEL, MOUSE or MUNKNOWN on failure

**2.25.3.4 bool Mouse::setLMRButton ( WPARAM *wParam* ) [protected]**

1 = left, 2 = middle, 3 = right, 4 = wheel, 0 = error

1. Author Robin Stenvi

**Parameters**

in	<i>wParam</i>	Says what type of mouse event that happened, foolows from the hook.
----	---------------	---

**Returns**

Returns true if we succeed, false if we fail, safe to continue regardless.

**2.25.3.5 void Mouse::setLLEvent ( WPARAM *wParam*, MSLHOOKSTRUCT *msll* )**

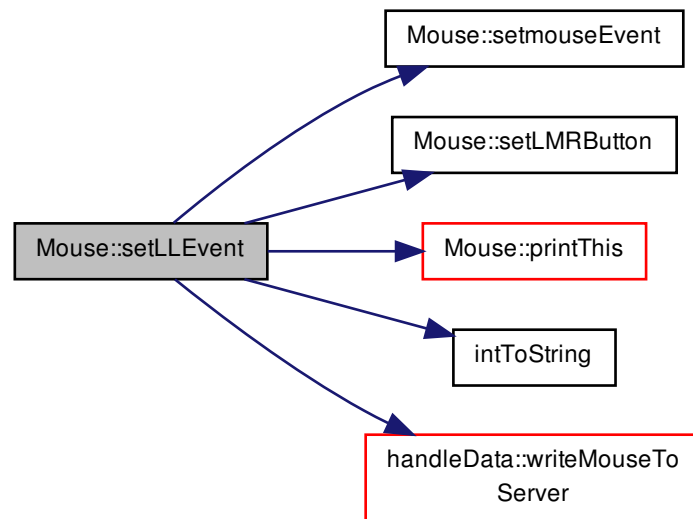
Sets all the appropriate data in a low level event and send it to the server.

1. Author Robin Stenvi

**Parameters**

in	<i>wParam</i>	Metadata about the mouse event (from Windows)
in	<i>msll</i>	Information about the mouse event (from Windows)

Here is the call graph for this function:

**2.26 MouseInfo Struct Reference**

All variable info we need to write about a mouse event.

**Public Attributes**

- UINT `Mtype`  
*Tells what type of event for filtering.*
- std::string `type`  
*D/U/M/W.*
- std::string `delta`  
*U/D or delta value.*
- std::string `mouseButton`

- 1/2/3/4 = I/R/M/W
- POINT `pt`
  - x, y coordinates*
- DWORD `timestamp`
  - timestamp*

### 2.26.1 Detailed Description

All variable info we need to write about a mouse event.

## 2.27 myWinEvent Class Reference

Implements MSAA functionality.

### Public Member Functions

- `myWinEvent` ()
  - Initialize all variables.*
- `~myWinEvent` ()
  - Empty destructor.*
- HRESULT `registerwinEvent` ()
  - Registers for all the events we want to receive.*
- std::string `getProcName` (HWND hwnd, DWORD threadId)
  - Gets a process name based on the thread ID for the process.*
- HRESULT `unregisterwinEvent` ()
  - Removed what we registered for earlier.*

### Static Public Member Functions

- static void `WinEventProc` (HWINETHEHOOK hook, DWORD event, HWND hwnd1, LONG id-Object, LONG idChild, DWORD dwEventThread, DWORD dwmsEventTime)
  - Callback function that is called when the events we registered for occur.*

### 2.27.1 Detailed Description

Implements MSAA functionality.

There are some events we don't retrieve from UIA, so we use MSAA to get those events.

1. Author Robin Stenvi

### 2.27.2 Constructor & Destructor Documentation

#### 2.27.2.1 myWinEvent::myWinEvent ( )

Initialize all variables.

1. Author Robin Stenvi

#### 2.27.2.2 myWinEvent::~myWinEvent ( )

Empty destructor.



1. Author Robin Stenvi

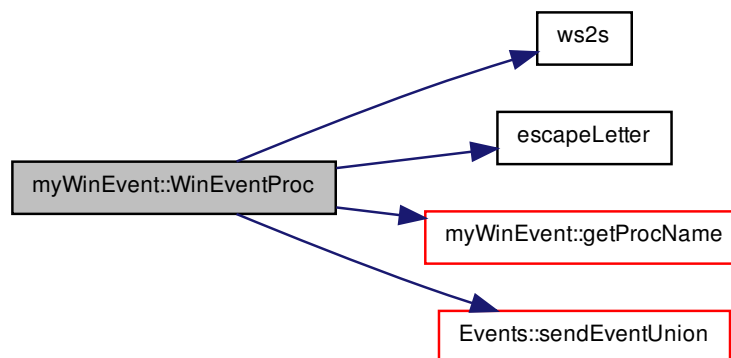
### 2.27.3 Member Function Documentation

**2.27.3.1** void myWinEvent::WinEventProc ( HWINEVENTHOOK *hook*, DWORD *event*, HWND *hwnd1*, LONG *idObject*, LONG *idChild*, DWORD *dwEventThread*, DWORD *dwmsEventTime* ) [static]

Callback function that is called when the events we registered for occur.

1. Author Robin Stenvi

Here is the call graph for this function:



**2.27.3.2** HRESULT myWinEvent::registerwinEvent ( )

Registers for all the events we want to receive.

1. Author Robin Stenvi

#### Returns

Returns S\_OK or E\_FAIL.

Here is the call graph for this function:



### 2.27.3.3 std::string myWinEvent::getProcName ( HWND *hwnd*, DWORD *threadId* )

Gets a process name based on the thread ID for the process.

It will add the process ID and name to the other list if it finds it. The function is also created so that we can retrieve the name of administrator processes without being administrator.

1. Author Robin Stenvi

#### Parameters

in	<i>hwnd</i>	The HWND to the application you want the name to.
in	<i>threadId</i>	The thread ID of a thread in the process.

#### Returns

Returns the name in the form of std::wstring.

Here is the call graph for this function:



### 2.27.3.4 HRESULT myWinEvent::unregisterwinEvent ( )

Removed what we registered for earlier.

1. Author Robin Stenvi

#### Returns

Returns S\_OK if we succeed.

## 2.28 processList Struct Reference

A list of processes so we can retrieve that information faster.

#### Public Attributes

- std::string [name](#)  
*The process executable name.*
- UINT [ID](#)  
*The process id we use when comparing.*

### 2.28.1 Detailed Description

A list of processes so we can retrieve that information faster.

## 2.29 sendData::progressRange Struct Reference

Holds the progress range of the progress element.

### 2.29.1 Detailed Description

Holds the progress range of the progress element.

## 2.30 propertyEventHandler Class Reference

Class that deals with all Property change events.

### Public Member Functions

- [propertyEventHandler](#) ()
- ULONG STDMETHODCALLTYPE [AddRef](#) ()
- ULONG STDMETHODCALLTYPE [Release](#) ()
- HRESULT STDMETHODCALLTYPE [QueryInterface](#) (REFIID riid, void \*\*ppvInterface)
- HRESULT STDMETHODCALLTYPE [HandlePropertyChangedEvent](#) (IUIAutomationElement \*pSender, PROPERTYID propertyID, VARIANT newValue)  
*The function that is called when a property event happens.*
- HRESULT [StartEventHandler](#) (HWND hDlg)  
*Start the event handlers, parameter should point to main dialog.*
- void [Uninitialize](#) ()  
*Here we tell the background thread to close down.*
- void [removeEventHandler](#) ()  
*Removes all event handlers and stops listening.*

### Protected Member Functions

- void [cleanup](#) ()  
*Release object that where created on this thread.*
- HRESULT [registerEventHandler](#) ()  
*Registers all the properties in cache and registers the events we are after.*

### Static Protected Member Functions

- static DWORD WINAPI [listenerThreadProc](#) (LPVOID lpParameter)  
*This is where the thread will run, listening for messages and take appropriate action.*

### Private Attributes

- LONG [\\_refCount](#)  
*Reference count.*
- IUIAutomation \* [automation](#)  
*UI Automation object we use the get the element tree.*
- IUIAutomationElement \* [rootElem](#)  
*Pointer to the root element in the UIA tree.*
- HWND [mainHwnd](#)

- Pointer to the main window, can be used to send messages back (currently not used)*
- HANDLE `backThreadHandle`  
*Handle to the worker thread we are creating to listen for events.*
- DWORD `backThread`  
*ID to the worker thread we are creating to listen for events.*
- HANDLE `eventListenerReady`  
*Handle to the event object.*
- BOOL `eventHandlerAdded`  
*If event handlers has been added or not.*
- IUIAutomationCacheRequest \* `cache`  
*Cache for faster retrieval of attributes.*

### 2.30.1 Detailed Description

Class that deals with all Property change events.

Receives an event if there is a visual change (VC) in the window. The cache has the following properties:

- Automation ID
- Element type (control type)
- The rectangle of each element
- Element type name (localized control type)
- Name
- Process ID

1. Author Robin Stenvi

#### Date

2012-01-21 - 2012-01-21 (Last modified)

### 2.30.2 Constructor & Destructor Documentation

#### 2.30.2.1 `propertyEventHandler::propertyEventHandler ( )`

1. Author Robin Stenvi

### 2.30.3 Member Function Documentation

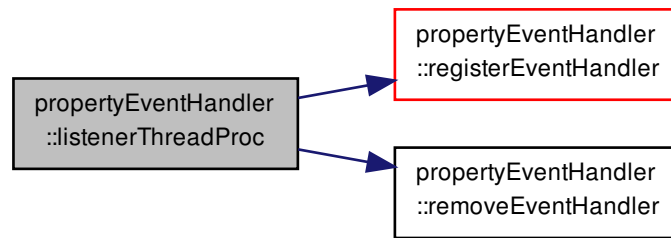
#### 2.30.3.1 `DWORD WINAPI propertyEventHandler::listenerThreadProc ( LPVOID lpParameter )` `[static]`, `[protected]`

This is where the thread will run, listening for messages and take appropriate action.

Message "WM\_BELT\_UITOEVENTHANDLER\_REGISTEREVENTHANDLER" registers the event handler. Message "WM\_BELT\_BACKTHREAD\_CLOSEDOWN2" closes the eventhandlers and shut down this thread

1. Author Robin Stenvi

Here is the call graph for this function:



### 2.30.3.2 void propertyEventHandler::cleanup ( ) [protected]

Release object that where created on this thread.

We must not remove the event handler here, it has to be removed on the same thread that added it.

1. Author Robin Stenvi

### 2.30.3.3 HRESULT propertyEventHandler::registerEventHandler ( ) [protected]

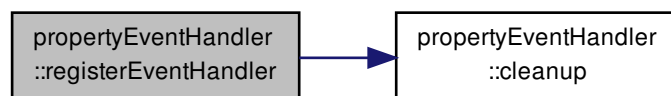
Registers all the properties in cache and registers the events we are after.

1. Author Robin Stenvi

#### Returns

Returns `S_OK` if we succeed, otherwise it returns the `HRESULT` from the Windows function where it failed.

Here is the call graph for this function:



### 2.30.3.4 ULONG STDMETHODCALLTYPE propertyEventHandler::AddRef ( )

1. Author Robin Stenvi

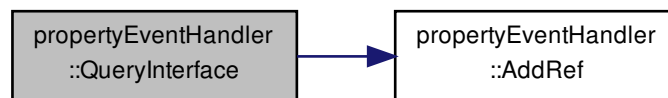
### 2.30.3.5 ULONG STDMETHODCALLTYPE propertyEventHandler::Release ( )

1. Author Robin Stenvi

### 2.30.3.6 HRESULT STDMETHODCALLTYPE propertyEventHandler::QueryInterface ( REFIID riid, void \*\* ppInterface )

1. Author Robin Stenvi

Here is the call graph for this function:

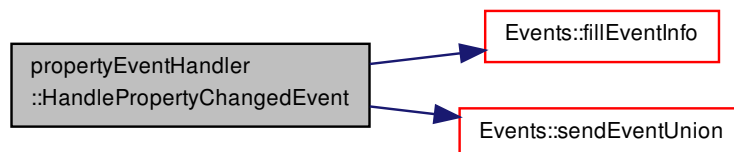


### 2.30.3.7 HRESULT STDMETHODCALLTYPE propertyEventHandler::HandlePropertyChangedEvent ( IUIAutomationElement \* pSender, PROPERTYID propertyID, VARIANT newValue )

The function that is called when a property event happens.

1. Author Robin Stenvi

Here is the call graph for this function:

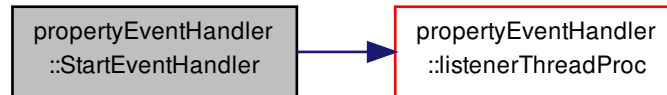


### 2.30.3.8 HRESULT propertyEventHandler::StartEventHandler ( HWND hDlg )

Start the event handlers, parameter should point to main dialog.

1. Author Robin Stenvi

Here is the call graph for this function:

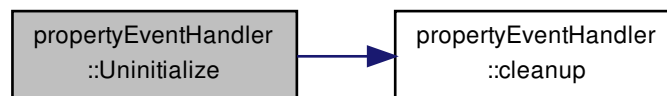


#### 2.30.3.9 void propertyEventHandler::Uninitialize ( )

Here we tell the background thread to close down.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.30.3.10 void propertyEventHandler::removeEventHandler ( )

Removes all event handlers and stops listening.

This have to be done on the same thread in which it was created

1. Author Robin Stenvi

## 2.31 Screen Struct Reference

Information about a physical screen.

### Public Attributes

- [HMONITOR ID](#)  
*Internal ID we use to recognize screen.*
- [DWORD time](#)  
*The time we detected the screen.*
- [LPRECT resolution](#)

- *The rectangle representing the screen.*
- int [num](#)  
*Which number in the list we are at.*
- bool [first](#)  
*If it is the first time we saw this.*

### 2.31.1 Detailed Description

Information about a physical screen.

1. Author Robin Stenvi

## 2.32 sendData Class Reference

Dialog that handles everything when user wants to send local file to server.

### Classes

- struct [Excluded](#)  
*Holds times that are excluded from the user.*
- struct [ExcludeIndex](#)  
*Which index the timestamps points to.*
- struct [File](#)  
*Holds all information we need to know about a file, is filled gradually.*
- struct [progressRange](#)  
*Holds the progress range of the progress element.*
- struct [Thread](#)  
*Information about each thread.*

### Public Member Functions

- [sendData](#) (CWnd \*pParent=NULL)  
*Construct that find metadata about all the files, so they are ready to be displayed.*
- virtual [~sendData](#) ()  
*Closes down each active thread.*
- LRESULT [OnGetDefID](#) (WPARAM wp, LPARAM lp)  
*Stop cancel from being the default button when enter is pressed.*
- bool [sendFileToServer](#) (int curr=-1)  
*Sends the selected file to server.*
- afx\_msg void [sendToServerButton](#) ()  
*Intermediary function that the button calls when the user want to send a session to the server.*
- afx\_msg void [OnLbnSelchangefileList](#) ()  
*Called when the user changes the file marked.*
- afx\_msg LRESULT [accFileDrop](#) (WPARAM w, LPARAM l)  
*The user can drag file to be included in the list.*
- afx\_msg void [undoTimeframeExclusion](#) ()  
*Removes a timeframe that has been added earlier.*
- afx\_msg void [filterTimeframeButon](#) ()  
*Filters a given timeframe that the user has given.*
- afx\_msg void [deleteFileButton](#) ()



*Wrapper for realDeleteFile, this function supply the number in our index.*

- `afx_msg void OnBnClickedCancel ()`  
*Overrides cancel button to only hide this window.*

### Protected Member Functions

- `void setProgress (const double value, int stat=0)`  
*Sets progress bar indicating how much is finished.*
- `virtual void DoDataExchange (CDataExchange *pDX)`  
*Initiates data exchange between elements.*
- `void addAllFiles ()`  
*Called each time the dialog is shown, initializes all the UI elements.*
- `bool readAllFiles ()`  
*Called on initialization, reads in all files to determine metadata we can show to the user.*
- `bool readFileLines (File &file)`  
*Reads in all the lines in a file and store it in the appropriate vector specified by currSel.*
- `std::wstring getLastLine (const std::wstring filename, int &length)`  
*Retrieves the last line in a file, and the length of the file in lines.*
- `void setTotalEvents (const int n)`  
*Sets the total number of events in a file.*
- `int binarySearch (const std::vector< std::wstring > arr, const SYSTEMTIME time, const bool before=true)`  
*Modified binary search to find the index before or after our time.*
- `bool isExcluded (const int n, File file)`  
*Checks if a current number is in the excluded list.*
- `void setCurrentFilter ()`  
*Resets the excludedListBox and sets the correct content based on currSel.*
- `void realDeleteFile (File file)`  
*Deletes a file from system and memory.*
- `void writeProgress (File files, int lines)`  
*Saves our current progress to file.*
- `void sendPreviousFile ()`  
*If we have unfinished files, this function will send the remaining files.*

### Static Protected Member Functions

- `static void sendFileToServerStatic (void *p)`  
*Static function that should run on it's own thread.*

### Private Attributes

- `std::vector< File > files`  
*Vector of all the available files.*
- `int currSel`  
*Which item is currently selected among in fileList, points to an index in files.*
- `CDateTimeCtrl startDate`  
*Time box so the user can choose what to exclude.*
- `CDateTimeCtrl stopDate`  
*Time box so the user can choose what to exclude.*
- `CEdit remainingEvents`

- Edit box so we can say to the user how many events are remaining.*

  - CListBox [excludedListBox](#)
    - Unsorted list box over excluded timestamps, each index correspond to our excluded vector.*
  - CListBox [fileList](#)
    - Unsorted List of timestamps displayed to the user, each timestamp corresponds to an index in allFiles which determines where that file is.*
  - CProgressCtrl [serverProgress](#)
    - Progress bar so we can indicate how much work remains.*
  - [progressRange](#) range
    - The range of our progress bar.*
  - int [currProgress](#)
    - If multiple threads are working, this should only be accessed with a mutex.*
  - CButton [cancelButton](#)
    - Hide button.*
  - [Thread](#) \* [threads](#)
    - Array of all the running threads.*
  - int [currThread](#)
    - Number of threads that are running.*
  - HANDLE [threadListMutex](#)
    - Only one thread is allowed to create or destroy threads at a time.*
  - CEdit [filesToServerInfo](#)
    - Edit field for displaying if we send data to the server or not.*

#### Static Private Attributes

- static const int [MAX\\_THREADS](#) = 20
  - Max number of allowed threads.*

#### 2.32.1 Detailed Description

Dialog that handles everything when user wants to send local file to server.

This is meant as an extra option for the user. If the user don't want to send events concurrently, but wants to decide later if the program gathered some sensitive information. The user will here be given the option to filter out events based on when it happened.

- When the object is allocated we find all the files that has been saved for future logging. We also read all the files to find out when they end and how many events they contain, but we do not store all of them as it might require a lot of memory.
- When the user shows the dialog, we add start/stop timestamps for each file so the user can see when they start and end. This is based on the work already done in the constructor.
- When the user select a file, we display how many events it contains, but we do not read it in. Because the file might be several MB big and it could take up a lot of RAM if we do this for all the files.
- If the user chooses to exclude a given timeframe we will just mark the timeframe as being excluded, but we don't know how many events that is, because we haven't read the file yet.
- If the user wants to undo a timeframe exclusion, we just remove the times previously added.
- When the user wants to send something to the server, we first read the entire file into memory. The we do a binary search on all the given timeframes to find out which indexes we should exclude. Then we send each included event to the server while maintaining the progress bar. Then the file is deleted in all places.

- If the user clicks that he wants to delete a file, it is removed from all places.

#### Remarks

There will be a problem if the user send multiple files, should be able to handle that, have to use arrays, where each can be the size of the number of files we have. We now this in the constructor so it is not a problem, just more work. Our use of currSel is now a little dangerous, the user can switch while the thread is running and then we would have a problem.

#### 2.32.2 work

It might be hard for the user to remember when he did some sensitive work. So this mechanism could be more clever and filter out everything that was written in a specific email for example. That would require some redesign and much more work.

1. Author Robin Stenvi

#### 2.32.3 Constructor & Destructor Documentation

##### 2.32.3.1 sendData::sendData ( CWnd \* pParent = NULL )

Construct that find metadata about all the files, so they are ready to be displayed.

1. Author Robin Stenvi

#### Parameters

in	<i>pParent</i>	Sent to parent.
----	----------------	-----------------

Here is the call graph for this function:



##### 2.32.3.2 sendData::~sendData ( ) [virtual]

Closes down each active thread.

1. Author Robin Stenvi

#### 2.32.4 Member Function Documentation

##### 2.32.4.1 void sendData::setProgress ( const double value, int stat = 0 ) [protected]

Sets progress bar indicating how much is finished.

1. Author Robin Stenvi

#### Parameters

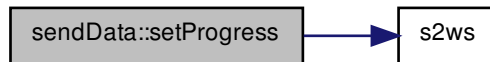
<i>in</i>	<i>value</i>	Progress made by the thread, measured in percent, so
<i>in</i>	<i>stat</i>	if it's the first time for the thread, 2 if it's the last time and 0 is default.

#### Remarks

If *stat* is different from 0, *value*, should be 0.

Produces some innaccurate results if we are sending multiple files with different lengths, or if the file has one or more lines excluded. But it does accomplish it's objective, which is to inform the user about the status.

Here is the call graph for this function:



#### 2.32.4.2 void sendData::DoDataExchange ( CDataExchange \* *pDX* ) [protected],[virtual]

Initiatess data exchange between elements.

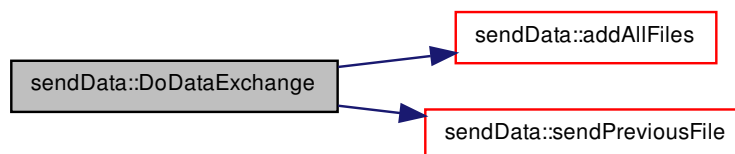
1. Author Robin Stenvi

Automatically generated

#### Parameters

<i>in</i>	<i>pDX</i>	Sent to CDialogEx::DoDataExchange().
-----------	------------	--------------------------------------

Here is the call graph for this function:



#### 2.32.4.3 void sendData::addAllFiles ( ) [protected]

Called each time the dialog is shown, initializes all the UI elements.

The object remembers previous sessions, so this function is called to restore everything to the way it was when the user left the last time. Is also called the first the dialog is shown.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.32.4.4 bool sendData::readAllFiles ( ) [protected]

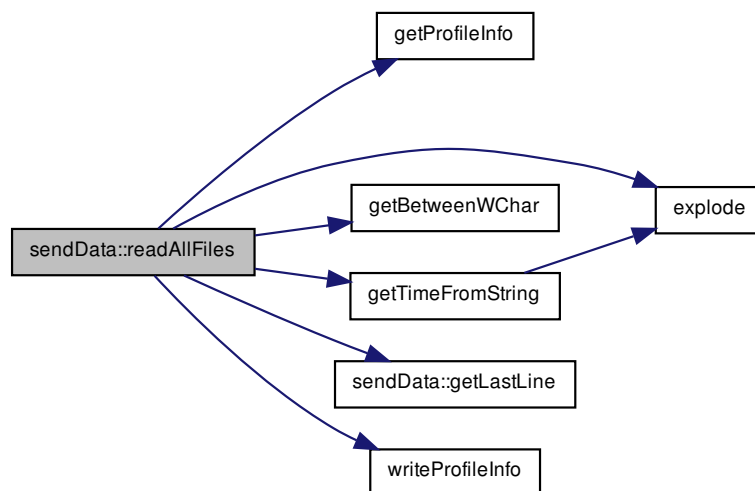
Called on initialization, reads in all files to determine metadata we can show to the user.

1. Author Robin Stenvi

#### Returns

Always returns true.

Here is the call graph for this function:



#### 2.32.4.5 bool sendData::readFileLines ( File & file ) [protected]

Reads in all the lines in a file and store it in the appropriate vector specified by `currSel`.

1. Author Robin Stenvi

#### Parameters

<i>in, out</i>	<i>file</i>	Should contain the filename and will contain all the lines afterwards.
----------------	-------------	--

#### Returns

Returns true if we succeed, otherwise false

#### 2.32.4.6 `std::wstring sendData::getLastLine ( const std::wstring filename, int & length )` [protected]

Retrieves the last line in a file, and the length of the file in lines.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>filename</i>	The file which should be read
<i>out</i>	<i>length</i>	The length of the file in lines.

#### Returns

Returns the last non-empty line of the file.

#### 2.32.4.7 `void sendData::setTotalEvents ( const int n )` [protected]

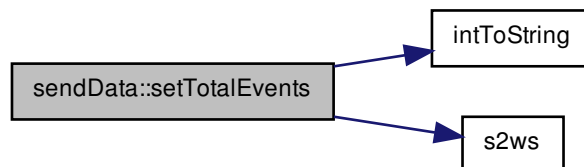
Sets the total number of events in a file.

1. Author Robin Stenvi

#### Parameters

<i>in</i>	<i>n</i>	The number that should be set
-----------	----------	-------------------------------

Here is the call graph for this function:



**2.32.4.8** `int sendData::binarySearch ( const std::vector< std::wstring > arr, const SYSTEMTIME time, const bool before = true )` [protected]

Modified binary search to find the index before or after our time.

1. Author Robin Stenvi

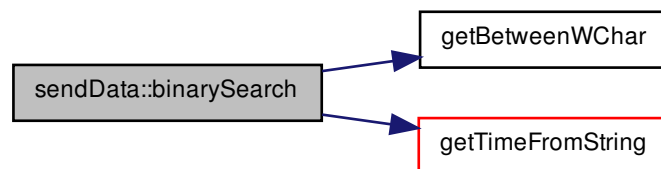
#### Parameters

in	<i>arr</i>	The vector with all the events.
in	<i>time</i>	The key you want to compare to
in	<i>before</i>	Says whether we are looking for the index before or after the time.

#### Returns

Returns the index to the first/last event that should be sent.

Here is the call graph for this function:



**2.32.4.9** `bool sendData::isExcluded ( const int n, File file )` [protected]

Checks if a current number is in the excluded list.

1. Author Robin Stenvi

#### Parameters

in	<i>n</i>	The array index
in	<i>file</i>	The file structure containing all the necessary information

#### Returns

Returns true if the event should not be written, false otherwise

**2.32.4.10** `void sendData::setCurrentFilter ( )` [protected]

Resets the excludedListBox and sets the correct content based on currSel.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.32.4.11 void sendData::sendFileToServerStatic ( void \* *p* ) [static],[protected]

Static function that should run on it's own thread.

1. Author Robin Stenvi

##### Parameters

<i>in</i>	<i>p</i>	pointer to an int that says the current index number or NULL to use default
-----------	----------	---

Here is the call graph for this function:



#### 2.32.4.12 void sendData::realDeleteFile ( File *file* ) [protected]

Deletes a file from system and memory.

1. Author Robin Stenvi

##### Parameters

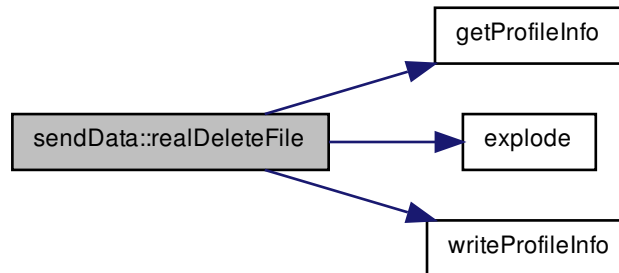
<i>in</i>	<i>file</i>	The that should be deleted
-----------	-------------	----------------------------



## Remarks

We need to use a mutex when deleting anything, because multiple threads might be running. This function will deal with the mutex, so this function is safe to call directly.

Here is the call graph for this function:



#### 2.32.4.13 void sendData::writeProgress ( File files, int lines ) [protected]

Saves our current progress to file.

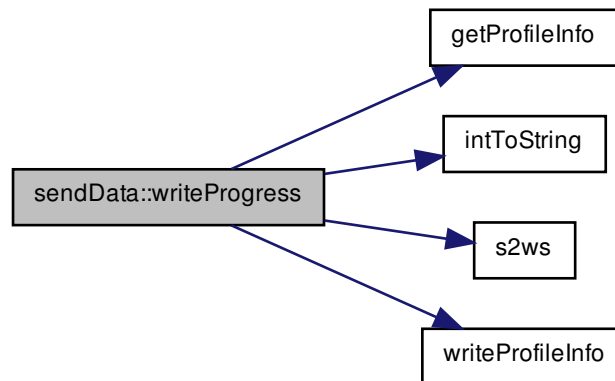
`openFiles` contains all the files we are currently working on but are not finished with. Each file information is separated by ";". We store three type of information about each file, the current line we are on and all the indexes we are going to exclude, each separated by ",", the start and stop index are separated by ":". So the file format is "file1,line1,start1:stop1...startN:stopN;file1...";

1. Author Robin Stenvi

## Parameters

in	<i>files</i>	Which file we are talking about
in	<i>lines</i>	How many lines have been sent

Here is the call graph for this function:

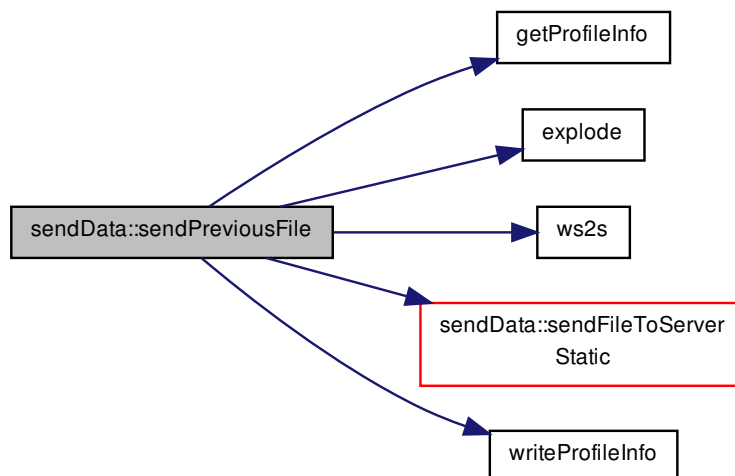


#### 2.32.4.14 void sendData::sendPreviousFile ( ) [protected]

If we have unfinished files, this function will send the remaining files.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.32.4.15 LRESULT sendData::OnGetDefID ( WPARAM wp, LPARAM lp )

Stop cancel from being the default button when enter is pressed.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>wp</i>	Unused.
<i>in</i>	<i>lp</i>	Unused.

**Returns**

Returns MAKELONG().

**2.32.4.16 bool sendData::sendFileToServer ( int *curr* = -1 )**

Sends the selected file to server.

This function will first read the entire file. Then it will use the excluded timestamps to find which indexes it should exclude, using the `binarySearch()`. It will then send all the events to server while maintaining the progress bar.

1. Author Robin Stenvi

**Parameters**

<i>in</i>	<i>curr</i>	The current index to our File array.
-----------	-------------	--------------------------------------

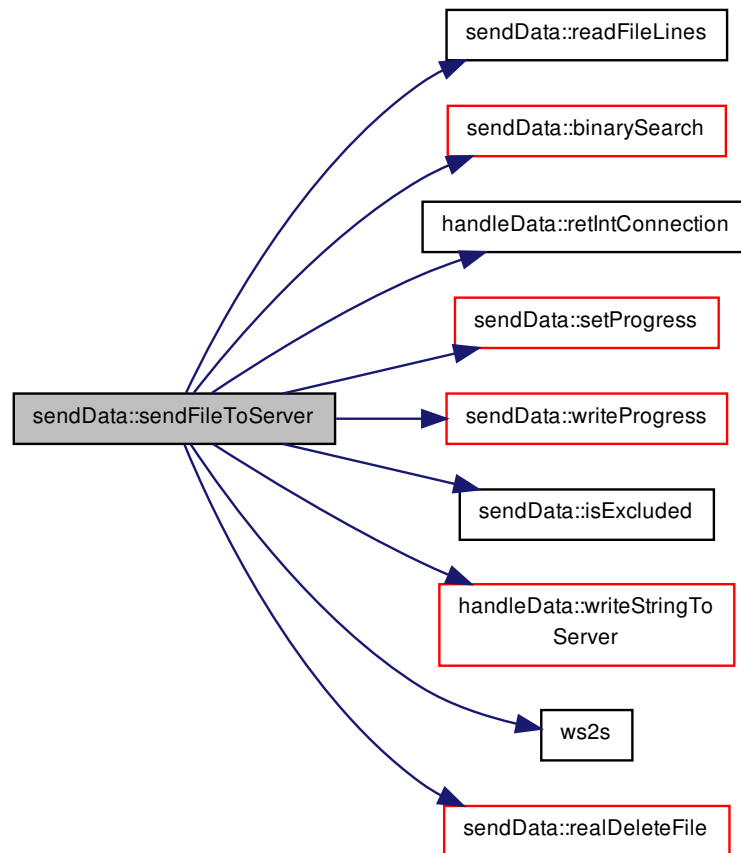
**Returns**

Returns true if we succeed, false otherwise

**Remarks**

This function can take a while and therefore it should be called on a separate thread so the user can do something else until it finishes. If the user wants to exit the application, this function will not exit. One solution might be to stop sending and only delete what we have sent so far. If we do that we should also store user configurations on that file and be sent automatically on next start-up.

Here is the call graph for this function:



#### 2.32.4.17 void sendData::sendToServerButton ( )

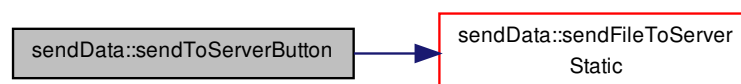
Intermediary function that the button calls when the user want to send a session to the server.

1. Author Robin Stenvi

#### Remarks

This function starts a new thread which again calls the actual function.

Here is the call graph for this function:



**2.32.4.18 void sendData::OnLbnSelchangefileList ( )**

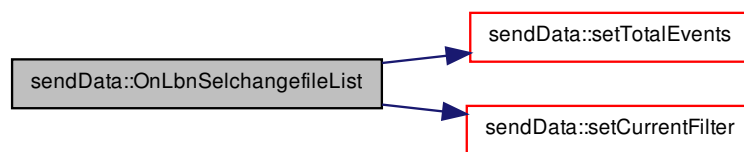
Called when the user changes the file marked.

1. Author Robin Stenvi

**Remarks**

We should change the timestamps to the start and stop accordingly, we also set number of events in that file.

Here is the call graph for this function:

**2.32.4.19 afx\_msg LRESULT sendData::accFileDrop ( WPARAM w, LPARAM l )**

The user can drag file to be included in the list.

1. Author Robin Stenvi

**Returns**

Always returns 0.

**Parameters**

<i>in</i>	<i>w</i>	Unused.
<i>in</i>	<i>l</i>	Unused.

**Remarks**

Does not work right now, need to make a separate function that jsut add one file, based on filename.

**2.32.4.20 void sendData::undoTimeframeExclusion ( )**

Removes a timeframe that has been added earlier.

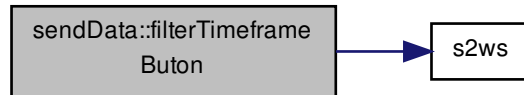
1. Author Robin Stenvi

**2.32.4.21 void sendData::filterTimeframeButon ( )**

Filters a given timeframe that the user has given.

1. Author Robin Stenvi

Here is the call graph for this function:

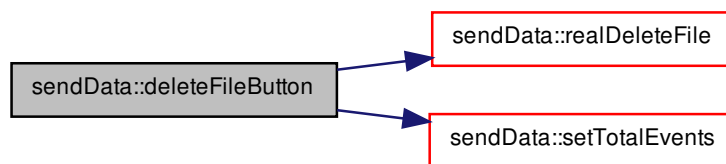


#### 2.32.4.22 void sendData::deleteFileButton ( )

Wrapper for `realDeleteFile`, this function supply the number in our index.

1. Author Robin Stenvi

Here is the call graph for this function:



#### 2.32.4.23 void sendData::OnBnClickedCancel ( )

Overrides cancel button to only hide this window.

1. Author Robin Stenvi

### 2.32.5 Member Data Documentation

#### 2.32.5.1 CListBox sendData::fileList [private]

Unsorted List of timestamps displayed to the user, each timestamp corresponds to an index in `allFiles` which determines where that file is.

## 2.33 SettingDialog Class Reference

A class for setting certain config settings of BeLT.

## Public Member Functions

- **SettingDialog** (CWnd \*pParent=NULL)  
*Constructor for the class.*
- virtual **~SettingDialog** ()  
*No deletes.*
- virtual INT\_PTR **DoModal** ()  
*Does Modal.*
- afx\_msg void **onSave** ()  
*Handler for the Save button in the dialog.*
- afx\_msg void **onStorage** ()  
*Updates the BOOL variable for automatic transmission of logged data.*
- afx\_msg void **onAutoUpdate** ()  
*Updates the BOOL variable for automatic updates of BeLT.*
- afx\_msg void **onAutoStart** ()  
*Updates the BOOL variable for automatic updates of BeLT.*
- afx\_msg void **advanced** ()  
*Show/hides certain fields based on a boolean variable.*
- afx\_msg void **reset** ()  
*Resets the settings in the dialog to the values in Settings.ini.*
- void **save** ()  
*saves the set data to the ini file when the save button is clocked*
- void **retrieveIniFile** ()  
*Retrieves the data from the profile.ini file.*
- BOOL **linkExists** ()  
*Checks if there is a file called BeLT.Ink in the startup folder.*
- BOOL **linkDeleted** ()  
*Deletes the shortcut to belt in the startup folder.*
- BOOL **linkCreated** ()  
*Creates a shortcut to BeLT in the startup folder.*
- BOOL **retAutoUpdate** ()  
*This returns the BOOL value of autoUpdate.*
- BOOL **retAutoStart** ()  
*This returns the BOOL value of autoStart.*
- std::wstring **retUpAddr** ()  
*Returns the wstring with the update servers address.*
- std::wstring **retLogAddr** ()  
*Returns an std::wstring with the log servers address.*
- int **retStorage** ()  
*This returns the BOOL value of autoTrans.*
- int **retUpPort** ()  
*Returns an int with the update servers HTTP port number.*
- int **retUpPortS** ()  
*Returns an int with the update servers HTTPS port number.*
- int **retLogPort** ()  
*Returns an int with the log servers port number.*
- BOOL **setAutoUpdate** (BOOL in)  
*This sets the BOOL value of autoStart.*
- BOOL **setAutoStart** (BOOL in)  
*This sets the BOOL value of autoStart.*
- BOOL **setUpAddr** (std::wstring in)

- This sets the wstring value of update address.*
- BOOL [setLogAddr](#) (std::wstring in)
 

*This sets the value of log address.*
- BOOL [setStorage](#) (int in)
 

*This sets the int value of storage.*
- BOOL [setUpPort](#) (int in, BOOL SSL)
 

*This sets the BOOL value of update port either the SSL or unencrypted.*
- BOOL [setLogPort](#) (int in)
 

*This sets the int value of the logging port.*

### Protected Member Functions

- virtual void [DoDataExchange](#) (CDataExchange \*pDX)
 

*Sets up data exchange and sets the text for the objects and sets the elements values.*

### Private Attributes

- std::wstring [logAddr](#)

*IP-address or domain name for the log server.*
- std::wstring [upAddr](#)

*IP-address or domain name for the update server.*
- int [logPort](#)

*Syslog-NG port number for the log server.*
- int [upPort](#)

*HTTP port number for the update server.*
- int [upPortS](#)

*HTTPS port number for the update server.*
- int [storage](#)

*Wether or not it should send to server, store locally or as CSV.*
- BOOL [autoStart](#)

*Wether or not it should start on boot.*
- BOOL [autoUpdate](#)

*Wether or not it should update automatically.*
- bool [dlgShow](#)

*Wether the dialog shows the advanced settings or normal settings.*
- std::wstring [warning](#)

*Warning text.*
- CEdit [IAddr](#)

*TextEdit field for log server address.*
- CEdit [IPort](#)

*TextEdit field for log server port Syslog-NG.*
- CEdit [uAddr](#)

*TextEdit field for update address.*
- CEdit [uPort](#)

*TextEdit field for update port HTTP.*
- CEdit [uPortS](#)

*TextEdit field for update port HTTPS.*
- CEdit [CText](#) [7]
 

*texts for the lables*
- CEdit [IWarning](#)



- Warning text field.*
- CButton [CButt](#) [2]
  - button for idok and cancel*
- CButton [aStore1](#)
  - Checkbox for controlling automatic sending of data to server.*
- CButton [aStore2](#)
  - Checkbox for controlling automatic storing data locally.*
- CButton [aStore3](#)
  - Checkbox for controlling automatic storing data as CSV.*
- CButton [aStart](#)
  - Checkbox for controlling automatic startup of BeLT.*
- CButton [aUpdate](#)
  - Checkbox for controlling automatic update of BeLT.*
- CButton [aAdvanced](#)
  - Checkbox for controlling showing advanced settings.*
- CButton [aDefault](#)
  - Checkbox for controlling resetting settings.*

### 2.33.1 Detailed Description

A class for setting certain config settings of BeLT.

This class reads data from the profile.ini and stores it in its variables. Otherwise it is set to default parameters. It handles all the configurations needed for setting server addresses/ports. It also controls whether or not BeLT should log automatically and update automatically

1. Author Magnus Øverbø - 25.03.2013

### 2.33.2 Constructor & Destructor Documentation

#### 2.33.2.1 SettingDialog::SettingDialog ( CWnd \* *pParent* = NULL )

Constructor for the class.

1. Author Magnus Øverbø - 25.03.2013

#### Parameters

in	<i>pParent</i>	Sent to parent.
----	----------------	-----------------

#### 2.33.2.2 SettingDialog::~SettingDialog ( ) [virtual]

No deletes.

1. Author Magnus Øverbø - 25.03.2013

### 2.33.3 Member Function Documentation

#### 2.33.3.1 void SettingDialog::DoDataExchange ( CDataExchange \* *pDX* ) [protected], [virtual]

Sets up data exchange and sets the text for the objects and sets the elements values.

retrieves data from the setting files and then sets the collected data to the GUI elements in the dialog

1. Author Magnus Øverbø - 25.03.2013

Magnus Øverbø - 05.04.2013

#### Parameters

in	<i>pDX</i>	Sent to CDialogEx::DoDataExchange().
----	------------	--------------------------------------

Here is the call graph for this function:



#### 2.33.3.2 INT\_PTR SettingDialog::DoModal ( ) [virtual]

Does Modal.

1. Author Magnus Øverbø - 25.03.2013

#### Returns

Returns CDialog::DoModal()

#### 2.33.3.3 void SettingDialog::onSave ( )

Handler for the Save button in the dialog.

1. Author Magnus Øverbø

Here is the call graph for this function:



#### 2.33.3.4 void SettingDialog::onStorage ( )

Updates the BOOL variable for automatic transmission of logged data.

1. Author Magnus Øverbø - 25.03.2013

#### 2.33.3.5 void SettingDialog::onAutoUpdate ( )

Updates the BOOL variable for automatic updates of BeLT.

This is run each time the checkbox i clicked

1. Author Magnus Øverbø - 27.03.2013

#### 2.33.3.6 void SettingDialog::onAutoStart ( )

Updates the BOOL variable for automatic updates of BeLT.

This is run each time the checkbox i clicked

1. Author Magnus Øverbø - 27.03.2013

#### 2.33.3.7 void SettingDialog::advanced ( )

Show/hides certain fields based on a boolean variable.

1. Author Magnus Øverbø - 06.05.2013

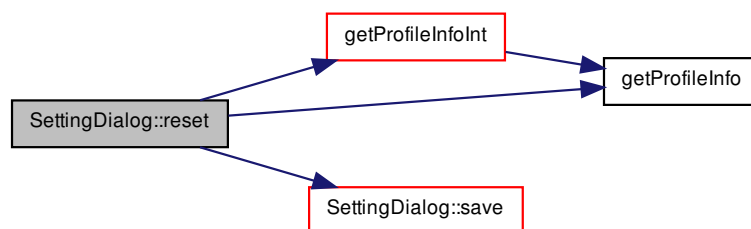
#### 2.33.3.8 void SettingDialog::reset ( )

Resets the settings in the dialog to the values in Settings.ini.

It retrieves and stores profile information using the strOperations functions and defaults to -1 if it isn't found

1. Author Magnus Øverbø

Here is the call graph for this function:



#### 2.33.3.9 void SettingDialog::save ( )

saves the set data to the ini file when the save button is clocked

It grabs the text from the interactive boxes and textfields. Then it writes them to the private profile "profile.ini" under the section server. Then it calls the EndDialog to exit the dialog cleanly. Saves the settings information as the following fields

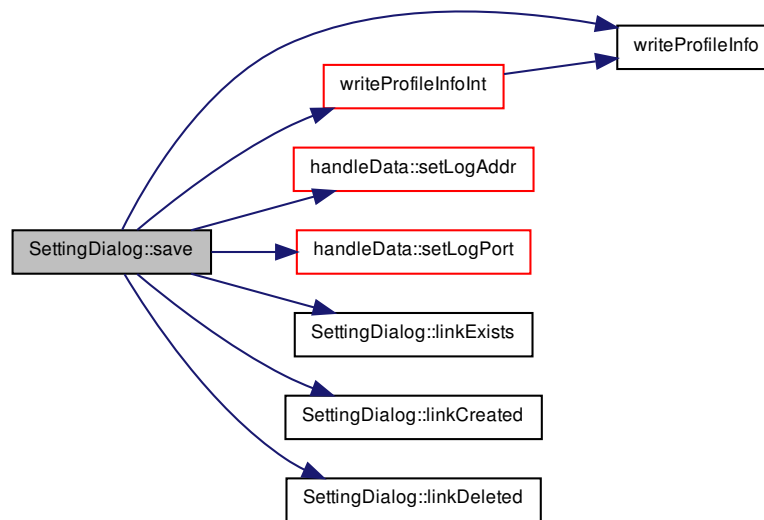
- serverPort - An int stating which port number to log to
- serverIP - A string stating the Syslog-NG IP/domain address to log to
- updatePort - An int stating which HTTP port number to the update server
- updatePortS - An int stating which HTTPS port number to the update server

- updateIP - A string stating the IP/domain address to log to
- autoUpdate - 2/1 which indicates OFF/ON
- autoSend - 2/1 which indicates OFF/ON
- autoStart - 2/1 which indicates OFF/ON

1. Author Magnus Øverbø - 25.03.2013

Magnus Øverbø - 19.04.2013

Here is the call graph for this function:



### 2.33.3.10 void SettingDialog::retrieveIniFile ( )

Retrieves the data from the profile.ini file.

Retrieves from profile.ini:

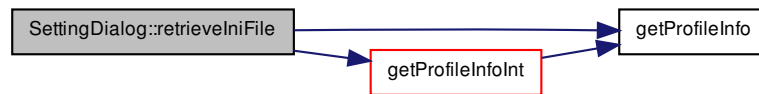
- serverPort - An int stating which port number to log to
- serverIP - A string stating the Syslog-NG IP/domain address to log to
- updatePort - An int stating which HTTP port number to the update server
- updatePortS - An int stating which HTTPS port number to the update server
- updateIP - A string stating the IP/domain address to log to
- autoUpdate - 2/1 which indicates OFF/ON
- autoStart - 2/1 which indicates OFF/ON
- storage - 1/2/3 which indicates SERVER/LOCAL/CSV

1. Author Magnus Øverbø - 25.03.2013

Magnus Øverbø - 11.04.2013

Magnus Øverbø - 05.04.2013

Here is the call graph for this function:



#### 2.33.3.11 BOOL SettingDialog::linkExists ( )

Checks if there is a file called BeLT.Ink in the startup folder.

1. Author Magnus Øverbø

##### Returns

TRUE if it is found and FALSE if the file or folder doesn't exist

#### 2.33.3.12 BOOL SettingDialog::linkDeleted ( )

Deletes the shortcut to belt in the startup folder.

1. Author Magnus Øverbø

##### Returns

TRUE if the deletion was successful and FALSE if the deletion failed

#### 2.33.3.13 BOOL SettingDialog::linkCreated ( )

Creates a shortcut to BeLT in the startup folder.

1. Author Magnus Øverbø

##### Returns

TRUE if it created the file and FALSE if it failed to create it

#### 2.33.3.14 BOOL SettingDialog::retAutoUpdate ( )

This returns the BOOL value of autoUpdate.

1. Author Magnus Øverbø - 27.03.2013

#### 2.33.3.15 BOOL SettingDialog::retAutoStart ( )

This returns the BOOL value of autoStart.

1. Author Magnus Øverbø - 27.03.2013

**Returns**

Returns TRUE if we are set to autostart, otherwise it returns false.

**2.33.3.16 std::wstring SettingDialog::retUpAddr ( )**

Returns the wstring with the update servers address.

1. Author Magnus Øverbø - 25.03.2013

**Returns**

Returns the address.

**2.33.3.17 std::wstring SettingDialog::retLogAddr ( )**

Returns an std::wstring with the log servers address.

1. Author Magnus Øverbø - 25.03.2013

**Returns**

Returns the string with the log address.

**2.33.3.18 int SettingDialog::retStorage ( )**

This returns the BOOL value of autoTrans.

1. Author Magnus Øverbø - 25.03.2013

**2.33.3.19 int SettingDialog::retUpPort ( )**

Returns an int with the update servers HTTP port number.

1. Author Magnus Øverbø - 25.03.2013

**Returns**

Returns the upgrade port.

**2.33.3.20 int SettingDialog::retUpPortS ( )**

Returns an int with the update servers HTTPS port number.

1. Author Magnus Øverbø - 05.04.2013

**Returns**

Returns the port number.

**2.33.3.21 int SettingDialog::retLogPort ( )**

Returns an int with the log servers port number.

1. Author Magnus Øverbø - 25.03.2013

#### Returns

Returns the log port.

#### 2.33.3.22 BOOL SettingDialog::setAutoUpdate ( BOOL *in* )

This sets the BOOL value of autoStart.

1. Author Magnus Øverbø - 19.04.2013

#### 2.33.3.23 BOOL SettingDialog::setAutoStart ( BOOL *in* )

This sets the BOOL value of autoStart.

1. Author Magnus Øverbø - 19.04.2013

#### 2.33.3.24 BOOL SettingDialog::setUpAddr ( std::wstring *in* )

This sets the wstring value of update address.

1. Author Magnus Øverbø - 19.04.2013

#### Parameters

<i>in</i>	<i>in</i>	The address.
-----------	-----------	--------------

#### Returns

Returns TRUE if we have valid address, otherwise it returns FALSE.

#### 2.33.3.25 BOOL SettingDialog::setLogAddr ( std::wstring *in* )

This sets the value of log address.

1. Author Magnus Øverbø - 19.04.2013

#### Parameters

<i>in</i>	<i>in</i>	The log address.
-----------	-----------	------------------

#### Returns

Returns TRUE if we have valid address, otherwise it returns FALSE.

#### 2.33.3.26 BOOL SettingDialog::setStorage ( int *in* )

This sets the int value of storage.

1. Author Magnus Øverbø - 19.04.2013

### 2.33.3.27 BOOL SettingDialog::setUpPort ( int in, BOOL SSL = TRUE )

This sets the BOOL value of update port either the SSL or unencrypted.

1. Author Magnus Øverbø - 19.04.2013

### 2.33.3.28 BOOL SettingDialog::setLogPort ( int in )

This sets the int value of the logging port.

1. Author Magnus Øverbø - 19.04.2013  
param[in] in The port number

#### Returns

Returns TRUE if we have valid port number, otherwise it returns FALSE.

## 2.34 Syslog1 Class Reference

Handles all the connection to the server.

#### Public Member Functions

- [Syslog1](#) ()  
*Initiate the TLS protocol, unique identifier and session number.*
- [~Syslog1](#) ()  
*Uninitializes and deletes variables used in the constructor.*
- int [startTls](#) ()  
*Initialize all the variables needed for TLS communication, also check if we have Internet connection.*
- int [sendMessage](#) (const std::string message)  
*Send one event to the server over TLS.*
- void [generateID](#) ()  
*Checks and generates the UID.*
- char \* [constructMessage](#) (Level level, Facility facility, DWORD TS, std::string appname, std::string procid, std::string msgid, std::string SD, std::string msg)  
*Format a message according to the syslog protocol.*
- bool [check\\_profile](#) ()  
*Validates the stored UID in profile.ini.*
- std::string [retID](#) ()  
*Returns the unique ID.*
- char \* [retSession](#) ()  
*Returns the session number we are at in the form of a char\*.*
- void [generateSID](#) ()  
*Generates and updates the session ID.*
- std::string [constructMessageStd](#) (Level level, Facility facility, DWORD TS, std::string appname, std::string procid, std::string msgid, std::string SD, std::string msg)  
*Format a message according to the syslog protocol.*
- void [setIpandPort](#) (std::wstring i=L"", int p=0)  
*Sets the correct IP-address and port number.*
- bool [initCom](#) ()  
*Initiates the COM library.*



### Protected Member Functions

- `std::string` [getHWID](#) (BSTR query, BSTR strClassProp)  
*Retrieves properties of the computer, used to generate unique ID.*
- `std::string` [getID](#) ()  
*Retrieves a random value from the computer motherboard, if the value is not random enough we generate our own.*

### Private Attributes

- `formatData` [format](#)  
*Helper class to format timestamps.*
- `std::string` [uniqueID](#)  
*Identificator for the user.*
- `std::wstring` [ip](#)  
*ip or url to the server*
- `int` [port](#)  
*Port number for logging.*
- `SOCKET` [sock](#)  
*The socket to our logging server.*
- `sockaddr_in` [sockaddrin](#)  
*The address to our server.*
- `char` [session](#) [22]  
*The current session number we are at.*
- `INetworkListManager *` [inMan](#)  
*To check if network is available.*
- `NLM_CONNECTIVITY` [conn](#)  
*Value to indicate current network status.*
- `bool` [initValid](#)  
*If we managed to initialize COM library.*

### Static Private Attributes

- `static const int` [SYSLOG\\_BUF\\_SIZE](#) = 2048  
*Max size for the syslog protocol.*

#### 2.34.1 Detailed Description

Handles all the connection to the server.

Sends all events using the syslog protocol (RFC 5424) over TLS using OpenSSL. Is also responsible for formatting the messages according to the syslog protocol.

1. Author Robin Stenvi

#### 2.34.2 Constructor & Destructor Documentation

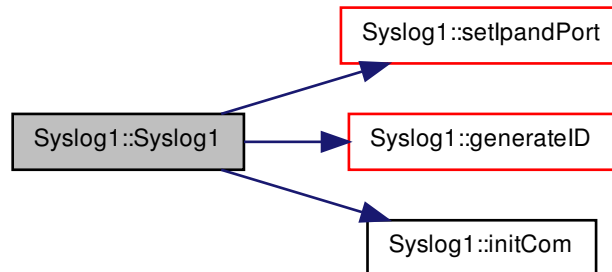
##### 2.34.2.1 Syslog1::Syslog1 ( )

Initiate the TLS protocol, unique identifier and session number.

1. Author Robin Stenvi

Magnus Øverbø

Here is the call graph for this function:



### 2.34.2.2 Syslog1::~~Syslog1 ( )

Uninitializes and deletes variables used in the constructor.

1. Author Robin Stenvi

### 2.34.3 Member Function Documentation

#### 2.34.3.1 std::string Syslog1::getHWID ( BSTR query, BSTR strClassProp ) [protected]

Retrieves properties of the computer, used to generate unique ID.

1. Author Robin Stenvi

#### Parameters

in	<i>query</i>	The query that shall be executed.
in	<i>strClassProp</i>	The property you want to retrieve

#### Returns

Returns an std::string representing the ID.

Here is the call graph for this function:



### 2.34.3.2 `std::string Syslog1::getID ( )` [protected]

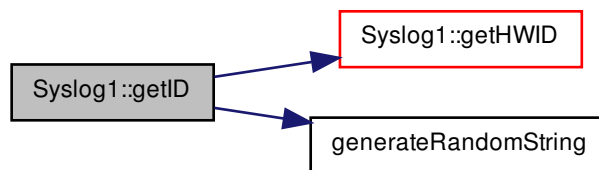
Retrieves a random value from the computer motherboard, if the value is not random enough we generate our own.

1. Author Robin Stenvi

#### Returns

Returns the random ID

Here is the call graph for this function:



### 2.34.3.3 `int Syslog1::startTls ( )`

Initialize all the variables needed for TLS communication, also check if we have Internet connection.

1. Author Robin Stenvi

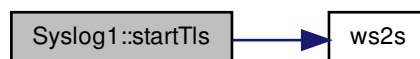
#### Remarks

An Internet connection is the only valid connection, so it will not work on an intranet, even though you might have access to the server.

#### Returns

Returns 0 if we succeed, any other value on failure

Here is the call graph for this function:



#### 2.34.3.4 int Syslog1::sendMessage ( const std::string *message* )

Send one event to the server over TLS.

It first checks if Internet is available, if it is it sends the message. If the server is on a local network and the user don't have an Internet connection it will not work, even though they technically have a connection to the server.

##### Parameters

<i>in</i>	<i>message</i>	The message that should be sent
-----------	----------------	---------------------------------

1. Author Robin Stenvi

##### Returns

Returns the number of character sent, should be equal to strlen(message), returns <= 0 on error

Here is the call graph for this function:



#### 2.34.3.5 void Syslog1::generateID ( )

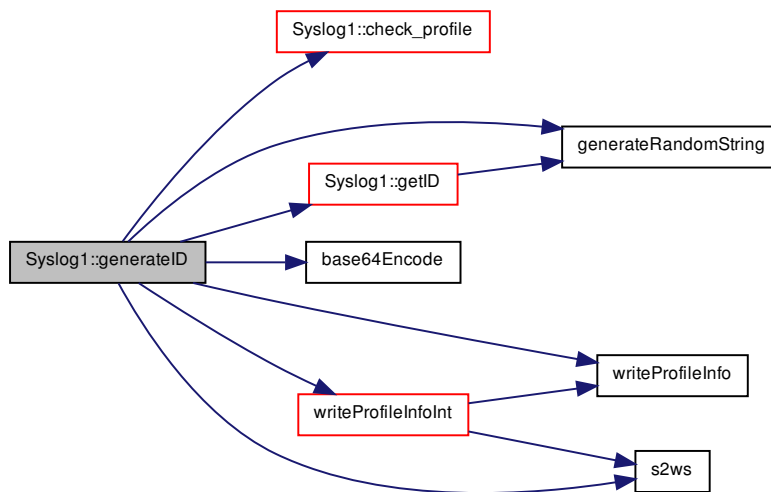
Checks and generates the UID.

First checks if the profile.ini content is valid otherwise it will create a new UID based on the logged in users username, unique to the current computer, and the HW information retrieved from GetID, which is unique to the world. If it fail to grab the username it will create 15 random chars instead, and if the HW id fails it is automatically generated a random ID inside the called function. Finally this is written to the profile.ini along with the session number 0.

1. Author Magnus Øverbø - 12.03.2013

Magnus Øverbø - 26.04.2013

Here is the call graph for this function:



**2.34.3.6** `char * Syslog1::constructMessage ( Level level, Facility facility, DWORD TS, std::string appname, std::string procid, std::string msgid, std::string SD, std::string msg )`

Format a message according to the syslog protocol.

1. Author Robin Stenvi

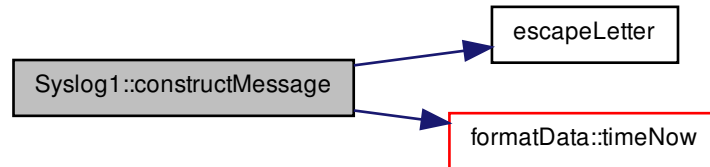
#### Parameters

in	<i>level</i>	The level of the message
in	<i>facility</i>	The facility of the message
in	<i>TS</i>	The timestamp of when the message occurred, should milliseconds since the system was started
in	<i>appname</i>	Application name
in	<i>procid</i>	Process ID
in	<i>msgid</i>	Identifier for what type of event occurred
in	<i>SD</i>	Structured data
in	<i>msg</i>	An extra message that can be sent

**Returns**

Returns a char\* array representing a valid syslog message

Here is the call graph for this function:

**2.34.3.7 bool Syslog1::check\_profile ( )**

Validates the stored UID in profile.ini.

Tries to create the Nislab/belt directory in appdata% dir. Then it tries to get the user id string from the file, if it's not 32char the function returns false. Previously if either directory doesn't exist it will return false also. The checking of returns from CreateDirectory and CreateFile should continue if the file/directory already exists.

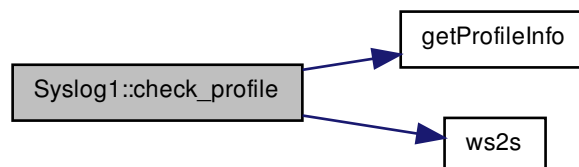
1. Author Magnus Øverbø - 12.03.2013

Magnus Øverbø - 26.04.2013

**Returns**

true if it finds a 32char long string under user->id and false if anything fails

Here is the call graph for this function:

**2.34.3.8 std::string Syslog1::retID ( )**

Returns the unique ID.

1. Author Robin Stenvi

#### Returns

Returns the ID.

#### 2.34.3.9 char \* Syslog1::retSession ( )

Returns the session number we are at in the form of a char\*.

1. Author Robin Stenvi

#### Returns

Returns the session number.

#### 2.34.3.10 void Syslog1::generateSID ( )

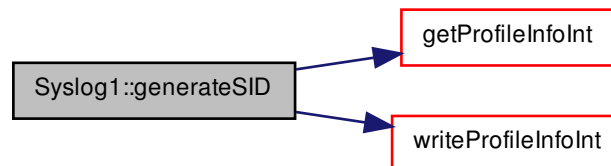
Generates and updates the session ID.

Reads the integer in Session from the profile.ini file and increments it and converts it to a string that is set for syslog class. If it is below or equal to zero the value is set to 1, which takes care of any wrapping that may occur. Then it writes the updated counter back to profile.ini.

1. Author Magnus Øverbø - 12.03.2013

Magnus Øverbø - 26.04.2013

Here is the call graph for this function:



#### 2.34.3.11 std::string Syslog1::constructMessageStd ( Level *level*, Facility *facility*, DWORD *TS*, std::string *appName*, std::string *procid*, std::string *msgid*, std::string *SD*, std::string *msg* )

Format a message according to the syslog protocol.

This function should make sure that our message comply with the Syslog protocol: <http://tools.ietf.org/html/rfc5424>

1. Author Robin Stenvi

**Parameters**

in	<i>level</i>	The level of the message
in	<i>facility</i>	The facility of the message
in	<i>TS</i>	The timestamp of when the message occurred, should milliseconds since the system was started
in	<i>appname</i>	Application name
in	<i>procid</i>	Process ID
in	<i>msgid</i>	Identifier for what type of event occurred
in	<i>SD</i>	Structured data
in	<i>msg</i>	An extra message that can be sent

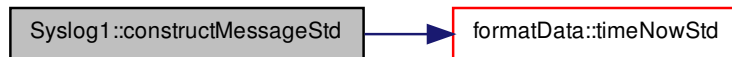
**Returns**

Returns an `std::string` representing a valid syslog message

**Remarks**

This function assumes that all variables are escaped and formatted properly, this is done to save time as we don't have to check all variables, some are generated by us.

Here is the call graph for this function:

**2.34.3.12 void Syslog1::setIpandPort ( std::wstring *i* = L" ", int *p* = 0 )**

Sets the correct IP-address and port number.

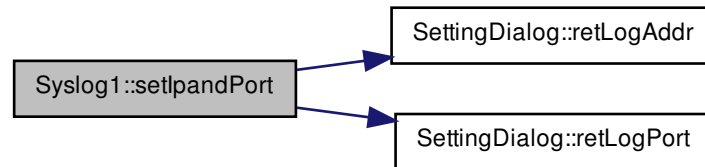
1. Author Robin Stenvi

**Parameters**

in	<i>i</i>	The IP-address, if the function should retrieve the IP-address dynamically, leave this empty, default value is empty
in	<i>p</i>	The port number, leave it as 0 (default) if we should find this dynamically.



Here is the call graph for this function:



### 2.34.3.13 bool Syslog1::initCom ( )

Initiates the COM library.

1. Author Robin Stenvi

#### Returns

Returns true if we succeed, false if we fail.

#### Remarks

When using multiple threads, this should be called before sending data.

## 2.35 sysResources Struct Reference

Contains information about system resources in use by the system, all values should be in percentage.

#### Public Attributes

- double `cpuLoad`  
*CPU load in percent.*
- DWORD `memLoad`  
*Memory load in percent.*
- DWORD `time`  
*The moment we did the measurement.*

### 2.35.1 Detailed Description

Contains information about system resources in use by the system, all values should be in percentage.

The time value is when we recorded the information.

## 2.36 sendData::Thread Struct Reference

Information about each thread.

**Public Attributes**

- DWORD [threadID](#)  
*The ID for the thread.*
- HANDLE [threadHandle](#)  
*The handle for the thread.*
- int [sel](#)  
*Which item in our list the thread is working on.*

**2.36.1 Detailed Description**

Information about each thread.

**2.37 UIAutomation Class Reference**

Handles the creation and destruction of all the UI Automation elements.

**Public Member Functions**

- [UIAutomation](#) ()  
*Creates variables and initializes them to NULL, does not start UI Automation.*
- void [cleanup](#) ()  
*Uninitializes all the data, releases all the eventhandlers.*
- bool [startEventHandlers](#) (HWND hDlg)  
*This is the function that main calls to start all events.*

**Private Attributes**

- [eventHandler](#) \* [events](#)  
*Class that is responsible for handling several event notifications.*
- [focusEventHandler](#) \* [focusEvents](#)  
*Class that is responsible for handling several event notifications.*
- [propertyEventHandler](#) \* [propertyEvents](#)  
*Class that is responsible for handling several event notifications.*

**2.37.1 Detailed Description**

Handles the creation and destruction of all the UI Automation elements.

This is the class that our main function deals with, this class handles all the other Automation objects, it creates them, initializes them and destroy them when we are finished.

1. Author Robin Stenvi - 2012-01-21

Robin Stenvi - 2012-01-21 (Last modified)

**2.37.2 Constructor & Destructor Documentation****2.37.2.1 UIAutomation::UIAutomation ( )**

Creates variables and initializes them to NULL, does not start UI Automation.

1. Author Robin Stenvi

### 2.37.3 Member Function Documentation

#### 2.37.3.1 void UIAutomation::cleanup ( )

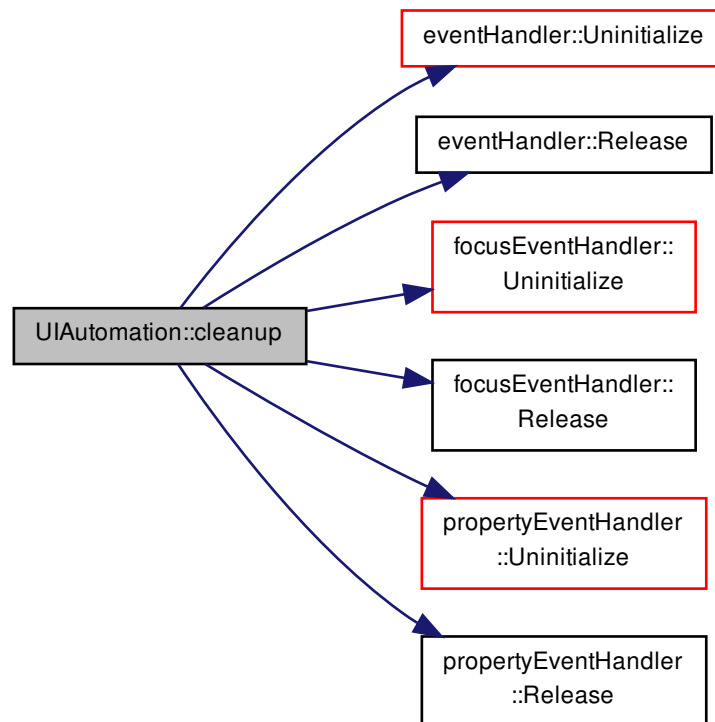
Uninitializes all the data, releases all the eventhandlers.

1. Author Robin Stenvi

#### Remarks

The objects deletes themselves, they should not be deleted here.

Here is the call graph for this function:



#### 2.37.3.2 bool UIAutomation::startEventHandlers ( HWND hDlg )

This is the function that main calls to start all events.

1. Author Robin Stenvi

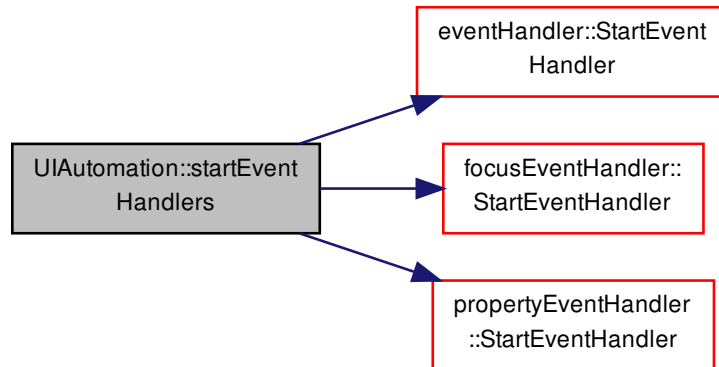
#### Parameters

<code>in</code>	<code>hDlg</code>	Handle to the main window
-----------------	-------------------	---------------------------

### Returns

A boolean value indicating if it failed or not, a false value means that it failed and none of the functions have started, it is the callers job to decide how it should be handled.

Here is the call graph for this function:



### 2.37.4 Member Data Documentation

#### 2.37.4.1 `eventHandler*` `UIAutomation::events` [private]

Class that is responsible for handling several event notifications.

#### 2.37.4.2 `focusEventHandler*` `UIAutomation::focusEvents` [private]

Class that is responsible for handling several event notifications.

#### 2.37.4.3 `propertyEventHandler*` `UIAutomation::propertyEvents` [private]

Class that is responsible for handling several event notifications.

## Index

- ~AboutDialog
  - AboutDialog, [34](#)
- ~CTrayNot
  - CTrayNot, [65](#)
- ~Cbelt\_mainDlg
  - Cbelt\_mainDlg, [40](#)
- ~Events
  - Events, [74](#)
- ~HWMonitor
  - HWMonitor, [140](#)
- ~Keylogger
  - Keylogger, [143](#)
- ~Mouse
  - Mouse, [148](#)
- ~SettingDialog
  - SettingDialog, [176](#)
- ~Syslog1
  - Syslog1, [185](#)
- ~eventHandler
  - eventHandler, [68](#)
- ~handleData
  - handleData, [100](#)
- ~myWinEvent
  - myWinEvent, [151](#)
- ~sendData
  - sendData, [162](#)
- \_\_declspec
  - Misc Global Functions, [3](#)
- AboutDialog, [34](#)
  - ~AboutDialog, [34](#)
  - AboutDialog, [34](#)
  - AboutDialog, [34](#)
  - DoDataExchange, [35](#)
  - DoModal, [35](#)
- accFileDrop
  - sendData, [172](#)
- accessKey
  - handleData, [125](#)
- addAllFiles
  - sendData, [163](#)
- AddRef
  - eventHandler, [69](#)
  - focusEventHandler, [90](#)
  - propertyEventHandler, [156](#)
- advanced
  - SettingDialog, [178](#)
- All enumerations, [5](#)
  - ERRORS, [5](#)
  - Facility, [5](#)
  - Level, [5](#)
- All the global / private structs, [19](#)
- All the key possible key variations, [20](#)
  - KEYBOARDFKEY, [23](#)
  - KEYBOARDNX, [23](#)
- AppendLine
  - Cbelt\_mainDlg, [43](#)
- base64Encode
  - Global function for string manipulation, [33](#)
- Bif flags for active system keys, [14](#)
- binarySearch
  - sendData, [165](#)
- Blacklist, [35](#)
- bstrToString
  - Global function for string manipulation, [26](#)
- CANCEL
  - Cbelt\_mainDlg, [41](#)
- CTrayNot, [64](#)
  - ~CTrayNot, [65](#)
  - CTrayNot, [65](#)
  - CTrayNot, [65](#)
  - SetState, [65](#)
- Cbelt\_mainApp, [35](#)
  - Cbelt\_mainApp, [36](#)
  - Cbelt\_mainApp, [36](#)
  - InitInstance, [36](#)
- Cbelt\_mainDlg, [36](#)
  - ~Cbelt\_mainDlg, [40](#)
  - AppendLine, [43](#)
  - CANCEL, [41](#)
  - Cbelt\_mainDlg, [40](#)
  - Cbelt\_mainDlg, [40](#)
  - changePause, [44](#)
  - controlListener, [53](#)
  - DoDataExchange, [40](#)
  - FAIL, [50](#)
  - manageRawInput, [43](#)
  - monitorHWUsage, [41](#)
  - OnAppAbout, [52](#)
  - OnAppExit, [47](#)
  - OnBnClickedButtonSendData, [57](#)
  - OnClose, [49](#)
  - OnDestroy, [46](#)
  - OnDeviceChange, [50](#)
  - OnEndSession, [48](#)
  - OnGetMinMaxInfo, [42](#)
  - OnHideapp, [46](#)
  - OnInitDialog, [42](#)
  - OnPaint, [45](#)
  - OnPowerState, [48](#)
  - OnQueryDragIcon, [45](#)
  - OnQueryEndSession, [47](#)
  - OnRawInput, [52](#)
  - OnSetDlg, [49](#)
  - OnSize, [47](#)
  - OnSysCommand, [45](#)
  - OnTrayNotify, [45](#)
  - OnTrayRestore, [46](#)

- pauseListener, 56
- printEvent, 44
- resumeListener, 57
- SUCCESS, 51
- sendKeyboardInfo, 43
- sendToServer, 54
- setFilterSettings, 54
- setIcon, 52
- startKeylog, 42
- startListener, 55
- stopListener, 53
- updateStat, 43
- changePause
  - Cbelt\_mainDlg, 44
- check
  - checkUpdate, 63
- check\_profile
  - Syslog1, 189
- checkScreen
  - focusEventHandler, 92
- checkUpdate, 57
  - check, 63
  - checkUpdate, 59
  - checkUpdate, 59
  - closeConnection, 62
  - closeTLS, 59
  - explode, 60
  - getFile, 61
  - getVersionNum, 60
  - getserverSettings, 60
  - initTLS, 59
  - initiateConnection, 61
  - internetAvailable, 63
  - newerVersion, 59
  - startTLS, 59
  - writeFile, 62
- cleanup
  - eventHandler, 69
  - focusEventHandler, 89
  - propertyEventHandler, 156
  - UIAutomation, 194
- clearButton
  - filterSettings, 86
- closeConnection
  - checkUpdate, 62
- closeCpuLoad
  - HWMonitor, 140
- closeTLS
  - checkUpdate, 59
- Constants for success or failure messages, 9
- constructMessage
  - Syslog1, 188
- constructMessageStd
  - Syslog1, 190
- controlListener
  - Cbelt\_mainDlg, 53
- cstringToString
  - Global function for string manipulation, 26
- Current mode of storage, 8
- Current server status at the client, 7
- Default Error Strings, 6
- deleteEventUnion
  - Events, 79
- deleteFileButton
  - sendData, 173
- deviceInfo, 66
- Diferent software events, 17
- difference
  - Mouse, 149
- Different colors for icons, 18
- Different colors used., 13
- DoDataExchange
  - AboutDialog, 35
  - Cbelt\_mainDlg, 40
  - sendData, 163
  - SettingDialog, 176
- DoModal
  - AboutDialog, 35
  - filterSettings, 86
  - SettingDialog, 177
- dwordToString
  - Global function for string manipulation, 25
- ERRORS
  - All enumerations, 5
- Each field in the list of patches, 16
- Each field in the update file, 15
- escapeKey
  - Keylogger, 144
- escapeLetter
  - Global function for string manipulation, 27
- eventHandler, 66
  - ~eventHandler, 68
  - AddRef, 69
  - cleanup, 69
  - eventHandler, 68
  - eventHandler, 68
  - HandleAutomationEvent, 70
  - listenerThreadProc, 68
  - QueryInterface, 70
  - registerEventHandler, 69
  - Release, 70
  - removeEventHandler, 69
  - StartEventHandler, 71
  - Uninitialize, 71
- eventInfoUnion, 72
- Events, 73
  - ~Events, 74
  - deleteEventUnion, 79
  - Events, 74
  - fillEventInfo, 78
  - getControlType, 76
  - getElemDescription, 75
  - getElemId, 77
  - getProcName, 74
  - getRectangle, 76

- getValueProperty, 75
- isDocument, 79
- isEdit, 79
- removeld, 77
- sendEventUnion, 77
- events
  - UIAutomation, 195
- Events from user, 11
- explode
  - checkUpdate, 60
  - Global function for string manipulation, 28
- FAIL
  - Cbelt\_mainDlg, 50
- Facility
  - All enumerations, 5
- fileList
  - sendData, 173
- fillEventInfo
  - Events, 78
- filterSettings, 81
  - clearButton, 86
  - DoModal, 86
  - filterSettings, 82
  - filterSettings, 82
  - keyPressAll, 85
  - keyPressDown, 85
  - keyPressUp, 85
  - MarkAllButton, 86
  - mouseAll, 83
  - mouseMove, 83
  - mousePress, 83
  - mousePressDown, 84
  - mousePressLeft, 84
  - mousePressMiddle, 84
  - mousePressRight, 85
  - mousePressUp, 84
  - mouseWheel, 83
  - retFilter, 86
  - setBoxes, 86
- filterTimeframeButon
  - sendData, 172
- findKeyDown
  - Keylogger, 144
- focusEventHandler, 87
  - AddRef, 90
  - checkScreen, 92
  - cleanup, 89
  - focusEventHandler, 88
  - focusEventHandler, 88
  - HandleFocusChangedEvent, 90
  - listenerThreadProc, 89
  - QueryInterface, 90
  - registerEventHandler, 89
  - Release, 90
  - removeEventHandler, 92
  - StartEventHandler, 91
  - Uninitialize, 91
- focusEvents
  - UIAutomation, 195
- formatData, 92
  - formatData, 93
  - formatData, 93
  - getRealTime, 93
  - timeNow, 94
  - timeNowStd, 94
- fromBstrToCString
  - Global function for string manipulation, 26
- generateID
  - Syslog1, 187
- generateRandomString
  - Global function for string manipulation, 32
- generateSID
  - Syslog1, 190
- getBeltToServer
  - handleData, 117
- getBetweenChar
  - Global function for string manipulation, 27
- getBetweenWChar
  - Global function for string manipulation, 28
- getControlType
  - Events, 76
- getCpuLoad
  - HWMonitor, 140
- getCsvBelt
  - handleData, 107
- getCsvDev
  - handleData, 105
- getCsvHID
  - handleData, 106
- getCsvHW
  - handleData, 104
- getCsvKey
  - handleData, 103
- getCsvKeyboard
  - handleData, 107
- getCsvMouse
  - handleData, 102
- getCsvRectangle
  - handleData, 108
- getCsvScreen
  - handleData, 105
- getCsvUIA
  - handleData, 103
- getDescSentenceKey
  - handleData, 119
- getDescSentenceMouse
  - handleData, 118
- getDevToServer
  - handleData, 115
- getElemDescription
  - Events, 75
- getElemId
  - Events, 77
- getEventToServer
  - handleData, 112
- getFile

- checkUpdate, 61
- getFormatHID
  - handleData, 111
- getFormatHW
  - handleData, 110
- getFormatKey
  - handleData, 109
- getFormatKeyboard
  - handleData, 110
- getFormatMouse
  - handleData, 108
- getFormatRectangle
  - handleData, 112
- getFormatScreen
  - handleData, 111
- getFormatUIA
  - handleData, 109
- getHIDToServer
  - handleData, 116
- getHWID
  - Syslog1, 185
- getHWToServer
  - handleData, 114
- getID
  - Syslog1, 185
- getKeyCount
  - Keylogger, 144
- getKeyToServer
  - handleData, 114
- getKeyboardToServer
  - handleData, 117
- getLastLine
  - sendData, 165
- getMemLoad
  - HWMonitor, 140
- getMouseToServer
  - handleData, 113
- getProcName
  - Events, 74
  - myWinEvent, 152
- getProfileInfo
  - Global function for string manipulation, 31
- getProfileInfoInt
  - Global function for string manipulation, 30
- getRealTime
  - formatData, 93
- getRectangle
  - Events, 76
- getScreenToServer
  - handleData, 116
- getTimeFromString
  - Global function for string manipulation, 29
- getTimeStamp
  - handleData, 101, 102
- getValueProperty
  - Events, 75
- getVersionNum
  - checkUpdate, 60
- getserverSettings
  - checkUpdate, 60
- Global function for string manipulation, 24
  - base64Encode, 33
  - bstrToString, 26
  - cstringToString, 26
  - dwordToString, 25
  - escapeLetter, 27
  - explode, 28
  - fromBstrToCstring, 26
  - generateRandomString, 32
  - getBetweenChar, 27
  - getBetweenWChar, 28
  - getProfileInfo, 31
  - getProfileInfoInt, 30
  - getTimeFromString, 29
  - intToString, 25
  - myAtoi, 27
  - openFile, 32
  - printUSASCII, 33
  - replaceLetter, 25
  - s2ws, 29
  - valueToString, 26
  - writeProfileInfo, 31
  - writeProfileInfoInt, 30
  - ws2s, 29
- Global variables, 2
- HIDDevice, 138
- HWMonitor, 139
  - ~HWMonitor, 140
  - closeCpuLoad, 140
  - getCpuLoad, 140
  - getMemLoad, 140
  - HWMonitor, 140
  - HWMonitor, 140
- HandleAutomationEvent
  - eventHandler, 70
- handleData, 95
  - ~handleData, 100
  - accessKey, 125
  - getBeltToServer, 117
  - getCsvBelt, 107
  - getCsvDev, 105
  - getCsvHID, 106
  - getCsvHW, 104
  - getCsvKey, 103
  - getCsvKeyboard, 107
  - getCsvMouse, 102
  - getCsvRectangle, 108
  - getCsvScreen, 105
  - getCsvUIA, 103
  - getDescSentenceKey, 119
  - getDescSentenceMouse, 118
  - getDevToServer, 115
  - getEventToServer, 112
  - getFormatHID, 111
  - getFormatHW, 110
  - getFormatKey, 109



- getFormatKeyboard, 110
- getFormatMouse, 108
- getFormatRectangle, 112
- getFormatScreen, 111
- getFormatUIA, 109
- getHIDToServer, 116
- getHWToServer, 114
- getKeyToServer, 114
- getKeyboardToServer, 117
- getMouseToServer, 113
- getScreenToServer, 116
- getTimestamp, 101, 102
- handleData, 100
- handleData, 100
- initCom, 137
- MAX\_STORAGE, 138
- retPaused, 136
- sendData, 120
- sendFullListToServer, 121
- sendFullListToServerStatic, 135
- sendListToServer, 120
- sendListToServerStatic, 135
- sendMissing, 122
- sendToServer, 119
- serverList, 138
- setLogAddr, 137
- setLogPort, 136
- startNewSession, 123
- stopCurrentSession, 124
- toggleServer, 125
- toggleServerStatic, 134
- updateSID, 136
- writeAll, 122
- writeData, 121
- writeDevToServer, 130
- writeEventToServer, 126
- writeHIDToServer, 132
- writeHWToServer, 129
- writeKeyToServer, 128
- writeKeyboardToServer, 133
- writeMissing, 121
- writeMissingStatic, 123
- writeMouseToServer, 127
- writeScreenToServer, 131
- writeStringToServer, 134
- writeTime, 125
- handleData::lastAll, 146
- lastInput, 147
- HandleFocusChangedEvent
  - focusEventHandler, 90
- HandlePropertyChangedEvent
  - propertyEventHandler, 157
- initCom
  - handleData, 137
  - Syslog1, 192
- InitInstance
  - Cbelt\_mainApp, 36
- initTLS
  - checkUpdate, 59
- initiateConnection
  - checkUpdate, 61
- intToString
  - Global function for string manipulation, 25
- internetAvailable
  - checkUpdate, 63
- isDocument
  - Events, 79
- isEdit
  - Events, 79
- isExcluded
  - sendData, 166
- KEYBOARDFKKEY
  - All the key possible key variations, 23
- KEYBOARDNX
  - All the key possible key variations, 23
- KeyInfo, 141
- keyPressAll
  - filterSettings, 85
- keyPressDown
  - filterSettings, 85
- keyPressUp
  - filterSettings, 85
- keyType, 146
  - KeyboardDevice, 141
- KeyboardDevice, 141
  - keyType, 141
  - lang, 141
- Keylogger, 142
  - ~Keylogger, 143
  - escapeKey, 144
  - findKeyDown, 144
  - getKeyCount, 144
  - Keylogger, 143
  - registerState, 146
  - retKeyevent, 143
  - setData, 143
  - setLLEvent, 145
  - setPassword, 145
- lang
  - KeyboardDevice, 141
- lastInput
  - handleData::lastAll, 147
- Level
  - All enumerations, 5
- linkCreated
  - SettingDialog, 180
- linkDeleted
  - SettingDialog, 180
- linkExists
  - SettingDialog, 180
- List of all classes, 1
- listenerThreadProc
  - eventHandler, 68
  - focusEventHandler, 89
  - propertyEventHandler, 155

- MAX\_STORAGE
  - handleData, 138
- manageRawInput
  - Cbelt\_mainDlg, 43
- MarkAllButton
  - filterSettings, 86
- Messages used throughout the application, 12
- Misc Global Functions, 3
  - \_\_declspec, 3
  - MyInfoEnumProc, 4
- monitorHWUsage
  - Cbelt\_mainDlg, 41
- Mouse, 147
  - ~Mouse, 148
  - difference, 149
  - Mouse, 148
  - printThis, 148
  - setLLEvent, 150
  - setLMRButton, 149
  - setMouseEvent, 149
- mouseAll
  - filterSettings, 83
- MouseInfo, 150
- mouseMove
  - filterSettings, 83
- mousePress
  - filterSettings, 83
- mousePressDown
  - filterSettings, 84
- mousePressLeft
  - filterSettings, 84
- mousePressMiddle
  - filterSettings, 84
- mousePressRight
  - filterSettings, 85
- mousePressUp
  - filterSettings, 84
- mouseWheel
  - filterSettings, 83
- myAtoi
  - Global function for string manipulation, 27
- MyInfoEnumProc
  - Misc Global Functions, 4
- myWinEvent, 151
  - ~myWinEvent, 151
  - getProcName, 152
  - myWinEvent, 151
  - myWinEvent, 151
  - registerwinEvent, 152
  - unregisterwinEvent, 153
  - WinEventProc, 152
- newerVersion
  - checkUpdate, 59
- OnAppAbout
  - Cbelt\_mainDlg, 52
- OnAppExit
  - Cbelt\_mainDlg, 47
- onAutoStart
  - SettingDialog, 178
- onAutoUpdate
  - SettingDialog, 177
- OnBnClickedButtonSendData
  - Cbelt\_mainDlg, 57
- OnBnClickedCancel
  - sendData, 173
- OnClose
  - Cbelt\_mainDlg, 49
- OnDestroy
  - Cbelt\_mainDlg, 46
- OnDeviceChange
  - Cbelt\_mainDlg, 50
- OnEndSession
  - Cbelt\_mainDlg, 48
- OnGetDefID
  - sendData, 169
- OnGetMinMaxInfo
  - Cbelt\_mainDlg, 42
- OnHideapp
  - Cbelt\_mainDlg, 46
- OnInitDialog
  - Cbelt\_mainDlg, 42
- OnLbnSelchangefileList
  - sendData, 172
- OnPaint
  - Cbelt\_mainDlg, 45
- OnPowerState
  - Cbelt\_mainDlg, 48
- OnQueryDragIcon
  - Cbelt\_mainDlg, 45
- OnQueryEndSession
  - Cbelt\_mainDlg, 47
- OnRawInput
  - Cbelt\_mainDlg, 52
- onSave
  - SettingDialog, 177
- OnSetDlg
  - Cbelt\_mainDlg, 49
- OnSize
  - Cbelt\_mainDlg, 47
- onStorage
  - SettingDialog, 177
- OnSysCommand
  - Cbelt\_mainDlg, 45
- OnTrayNotify
  - Cbelt\_mainDlg, 45
- OnTrayRestore
  - Cbelt\_mainDlg, 46
- openFile
  - Global function for string manipulation, 32
- pauseListener
  - Cbelt\_mainDlg, 56
- printEvent
  - Cbelt\_mainDlg, 44
- printThis
  - Mouse, 148

- printUSASCII
    - Global function for string manipulation, 33
  - processList, 153
  - propertyEventHandler, 154
    - AddRef, 156
    - cleanup, 156
    - HandlePropertyChangedEvent, 157
    - listenerThreadProc, 155
    - propertyEventHandler, 155
    - propertyEventHandler, 155
    - QueryInterface, 157
    - registerEventHandler, 156
    - Release, 157
    - removeEventHandler, 158
    - StartEventHandler, 157
    - Uninitialize, 158
  - propertyEvents
    - UIAutomation, 195
  - QueryInterface
    - eventHandler, 70
    - focusEventHandler, 90
    - propertyEventHandler, 157
  - readAllFiles
    - sendData, 164
  - readFileLines
    - sendData, 164
  - realDeleteFile
    - sendData, 167
  - registerEventHandler
    - eventHandler, 69
    - focusEventHandler, 89
    - propertyEventHandler, 156
  - registerState
    - Keylogger, 146
  - registerwinEvent
    - myWinEvent, 152
  - Release
    - eventHandler, 70
    - focusEventHandler, 90
    - propertyEventHandler, 157
  - removeEventHandler
    - eventHandler, 69
    - focusEventHandler, 92
    - propertyEventHandler, 158
  - removeld
    - Events, 77
  - replaceLetter
    - Global function for string manipulation, 25
  - reset
    - SettingDialog, 178
  - resumeListener
    - Cbelt\_mainDlg, 57
  - retAutoStart
    - SettingDialog, 180
  - retAutoUpdate
    - SettingDialog, 180
  - retFilter
    - filterSettings, 86
  - retID
    - Syslog1, 189
  - retKeyevent
    - Keylogger, 143
  - retLogAddr
    - SettingDialog, 181
  - retLogPort
    - SettingDialog, 181
  - retPaused
    - handleData, 136
  - retSession
    - Syslog1, 190
  - retStorage
    - SettingDialog, 181
  - retUpAddr
    - SettingDialog, 181
  - retUpPort
    - SettingDialog, 181
  - retUpPortS
    - SettingDialog, 181
  - retrieveIniFile
    - SettingDialog, 179
- s2ws
    - Global function for string manipulation, 29
  - SUCCESS
    - Cbelt\_mainDlg, 51
  - save
    - SettingDialog, 178
  - Screen, 158
  - sendData, 159
    - ~sendData, 162
    - accFileDrop, 172
    - addAllFiles, 163
    - binarySearch, 165
    - deleteFileButton, 173
    - DoDataExchange, 163
    - fileList, 173
    - filterTimeframeButon, 172
    - getLastLine, 165
    - handleData, 120
    - isExcluded, 166
    - OnBnClickedCancel, 173
    - OnGetDefID, 169
    - OnLbnSelchangefileList, 172
    - readAllFiles, 164
    - readFileLines, 164
    - realDeleteFile, 167
    - sendData, 162
    - sendFileToServer, 170
    - sendFileToServerStatic, 167
    - sendPreviousFile, 169
    - sendToServerButton, 171
    - sendData, 162
    - setCurrentFilter, 166
    - setProgress, 162
    - setTotalEvents, 165
    - undoTimeframeExclusion, 172
- 
- Generated with Doxygen on May 15, 2013

- writeProgress, 168
- sendData::ExcludeIndex, 80
- sendData::Excluded, 80
- sendData::File, 80
- sendData::Thread, 192
- sendData::progressRange, 154
- sendEventUnion
  - Events, 77
- sendFileToServer
  - sendData, 170
- sendFileToServerStatic
  - sendData, 167
- sendFullListToServer
  - handleData, 121
- sendFullListToServerStatic
  - handleData, 135
- sendKeyboardInfo
  - Cbelt\_mainDlg, 43
- sendListToServer
  - handleData, 120
- sendListToServerStatic
  - handleData, 135
- sendMessage
  - Syslog1, 186
- sendMissing
  - handleData, 122
- sendPreviousFile
  - sendData, 169
- sendToServer
  - Cbelt\_mainDlg, 54
  - handleData, 119
- sendToServerButton
  - sendData, 171
- serverList
  - handleData, 138
- setAutoStart
  - SettingDialog, 182
- setAutoUpdate
  - SettingDialog, 182
- setBoxes
  - filterSettings, 86
- setCurrentFilter
  - sendData, 166
- setData
  - Keylogger, 143
- setFilterSettings
  - Cbelt\_mainDlg, 54
- setIcon
  - Cbelt\_mainDlg, 52
- setIpandPort
  - Syslog1, 191
- setLLEvent
  - Keylogger, 145
  - Mouse, 150
- setLMRButton
  - Mouse, 149
- setLogAddr
  - handleData, 137
- SettingDialog, 182
- setLogPort
  - handleData, 136
  - SettingDialog, 183
- setPassword
  - Keylogger, 145
- setProgress
  - sendData, 162
- SetState
  - CTrayNot, 65
- setStorage
  - SettingDialog, 182
- setTotalEvents
  - sendData, 165
- setUpAddr
  - SettingDialog, 182
- setUpPort
  - SettingDialog, 183
- setMouseEvent
  - Mouse, 149
- SettingDialog, 173
  - ~SettingDialog, 176
  - advanced, 178
  - DoDataExchange, 176
  - DoModal, 177
  - linkCreated, 180
  - linkDeleted, 180
  - linkExists, 180
  - onAutoStart, 178
  - onAutoUpdate, 177
  - onSave, 177
  - onStorage, 177
  - reset, 178
  - retAutoStart, 180
  - retAutoUpdate, 180
  - retLogAddr, 181
  - retLogPort, 181
  - retStorage, 181
  - retUpAddr, 181
  - retUpPort, 181
  - retUpPortS, 181
  - retrievalniFile, 179
  - save, 178
  - setAutoStart, 182
  - setAutoUpdate, 182
  - setLogAddr, 182
  - setLogPort, 183
  - setStorage, 182
  - setUpAddr, 182
  - setUpPort, 183
  - SettingDialog, 176
  - SettingDialog, 176
- StartEventHandler
  - eventHandler, 71
  - focusEventHandler, 91
  - propertyEventHandler, 157
- startEventHandlers
  - UIAutomation, 194

- startKeylog
  - Cbelt\_mainDlg, 42
- startListener
  - Cbelt\_mainDlg, 55
- startNewSession
  - handleData, 123
- startTLS
  - checkUpdate, 59
- startTls
  - Syslog1, 186
- stopCurrentSession
  - handleData, 124
- stopListener
  - Cbelt\_mainDlg, 53
- sysResources, 192
- Syslog1, 183
  - ~Syslog1, 185
  - check\_profile, 189
  - constructMessage, 188
  - constructMessageStd, 190
  - generateID, 187
  - generateSID, 190
  - getHWID, 185
  - getID, 185
  - initCom, 192
  - retID, 189
  - retSession, 190
  - sendMessage, 186
  - setIpandPort, 191
  - startTls, 186
  - Syslog1, 184
- timeNow
  - formatData, 94
- timeNowStd
  - formatData, 94
- toggleServer
  - handleData, 125
- toggleServerStatic
  - handleData, 134
- UIAutomation, 193
  - cleanup, 194
  - events, 195
  - focusEvents, 195
  - propertyEvents, 195
  - startEventHandlers, 194
  - UIAutomation, 193
  - UIAutomation, 193
- undoTimeframeExclusion
  - sendData, 172
- Uninitialize
  - eventHandler, 71
  - focusEventHandler, 91
  - propertyEventHandler, 158
- unregisterwinEvent
  - myWinEvent, 153
- updateSID
  - handleData, 136
- updateStat
  - Cbelt\_mainDlg, 43
- valueToString
  - Global function for string manipulation, 26
- Which log event occurred, 10
- WinEventProc
  - myWinEvent, 152
- writeAll
  - handleData, 122
- writeData
  - handleData, 121
- writeDevToServer
  - handleData, 130
- writeEventToServer
  - handleData, 126
- writeFile
  - checkUpdate, 62
- writeHIDToServer
  - handleData, 132
- writeHWToServer
  - handleData, 129
- writeKeyToServer
  - handleData, 128
- writeKeyboardToServer
  - handleData, 133
- writeMissing
  - handleData, 121
- writeMissingStatic
  - handleData, 123
- writeMouseToServer
  - handleData, 127
- writeProfileInfo
  - Global function for string manipulation, 31
- writeProfileInfoInt
  - Global function for string manipulation, 30
- writeProgress
  - sendData, 168
- writeScreenToServer
  - handleData, 131
- writeStringToServer
  - handleData, 134
- writeTime
  - handleData, 125
- ws2s
  - Global function for string manipulation, 29



## **EULA - BeLT**

### End User Licence Agreement for Behaviour Logging Tool

#### **Table of content**

1. Licence agreement
2. Disclaimer
3. Intellectual Property rights
4. General information
5. About BeLT

#### **1) Licence Agreement**

1. All rights for BeLT is reserved to NISlab represented by Prof. Patrick Bours and Soumik Mondal
2. By installing this application you agree to, and promise you have read and understood the EULA, and accepted all its terms.
3. By agreeing to the licence agreement you are allowing BeLT to log and store information about you.
4. The user has the right to know what kind of information is being stored by BeLT and what its purpose is.
5. Any question may be directed at Prof. Patrick Bours

#### **2) Disclaimer**

1. BeLT **WILL** capture and store sensitive information about the user.
2. It is the users own responsibility to hinder the compromise of sensitive information when using BeLT
3. BeLT nor NISlab can be held liable for any trouble, loss or damage caused by BeLT or its information.

#### **3) Intellectual Property rights**

1. The code is owned and maintained by NISlab. Redistribution, use or derivation of this applicaiton, its source code or any other information related to BeLT is prohibited without NISlabs written consent.
2. BeLT is entirely owned by NISlab
3. Project maintainers are:  
Prof. Patrick Bours [patrick.bours@hig.no](mailto:patrick.bours@hig.no)  
Soumik Mondal [soumik.mondal@hig.no](mailto:soumik.mondal@hig.no)
4. The developers are:  
Robin Stenvi [robin.stenvi@hig.no](mailto:robin.stenvi@hig.no)  
Magnus Øverbø [magnus.Overbo@hig.no](mailto:magnus.Overbo@hig.no)  
Lasse T. Johansen [lasse.johansen@hig.no](mailto:lasse.johansen@hig.no)

#### **4) General Information**

1. BeLT was developed as a Bachelor Thesis in the spring of 2013
2. Errors can be reported to Soumik Mondal or Patrick Bours of NISlab, ([www.nislab.no](http://www.nislab.no))

#### **5) About BeLT**

1. BeLT captures and stores information about the users interaction with the computer.
2. BeLT captures the following:

1. Mouse interactions
  2. Keyboard interactions
  3. Hardware events/changes
  4. Software events/changes
  5. System events/changes
3. BeLT will transmit the data it captures to a centralised storage facility managed by NISlab
  4. The application comes with a user manual available on-line or by request from the project managers.



## G Work Log

### G.1 Work activity documentation

In figure 1 we have summarized our work activity and efforts, as a group and individuals.

Throughout our project we have worked consistently with good effort and without breaks, except two days during Easter. We have certain spikes in our activity which is a result of special events that occurred during our development process. The purple area in the background is the combined work effort of the group. The area represents the average of our combined efforts over the previous three days.

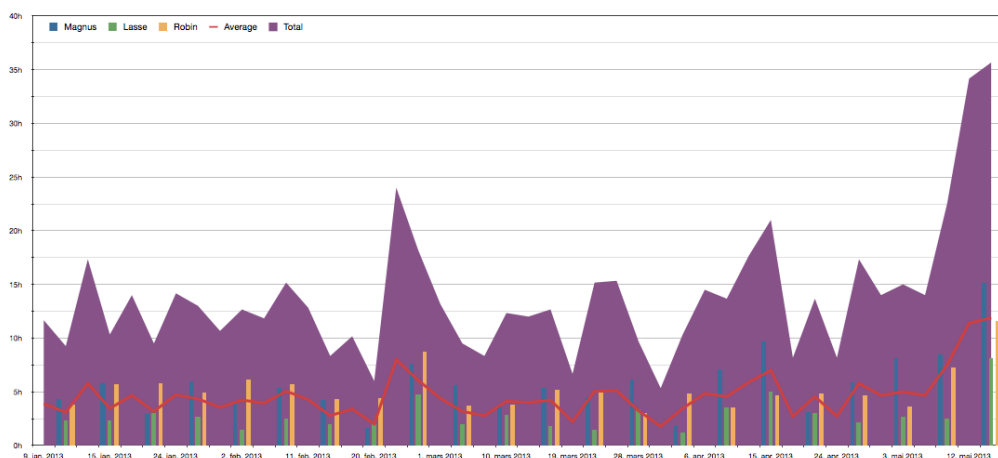


Figure 1: Graph of summarized work effort

It clearly shows that we've had certain points in our development when we've had a lot to do, but generally was our activity stable. Our first spike is during our first release of BeLT on the 14th of January. At this point we had to set up our environment, develop the basis of our graphical user interface and main application functionality. After this point our work effort was pretty steady until the week leading up to our server test. The time leading up to our server test was lower because we had done most of our development at this point and had to test our server communication before continuing. As a result of this we spent this time focusing on our additional courses and projects. Over the course of the weekend of the 22nd-24th of February we have a very high spike due to our testing phase.

Following the server test we had a lowered performance while doing the task of analysing and documenting our test. Then on the week before Easter we had an increase in our efforts when trying to create a driver for our application.

During the Easter holiday we have a rather large drop as caused by us taking the weekend off and resting before starting up again on the 1st of April with a group meeting. After this we had an increase in activity as since we had a rush on finishing our application, file format, database storage/export and starting our migration to a production system. Then on the 15th we finished most of our functionality in BeLT, the server

applications and main part of our tasks. Then some time got wasted waiting for a server which weren't allowed to use in the end which made us have a massive spike at the end during our production of our report and final version of BeLT.

The purple area in the background is our collaborative effort averaged over three days iterations. This means that this graph has one datapoint for each three days passed during our project. The red line is our groups average work efforts for all three members, averaged over three days iterations. This, along with our columns, representing the group members, shows how much effort was put into the project as opposed to the average work effort.

## G.2 Progress log

Progress log	
Date	Description
4/1-13	First meeting with counselor after project had started
10/1-13	First draft of preliminary project is finished and delivered supervisor.
11/1-13	Crash course with Tom Røise. Finished prototype to show what type of information is collected, includes keylogger, mouselogger and very limited UI Automation.
14/1-13	First meeting with the employer after the project has started. Defined the scope of the task and demonstrated prototype.
17/1-13	Created the first GUI prototype.
26/1-13	Released BeLT 0.1.0, with GUI to start, stop, pause and resume logging, could also display logged data to screen and write logged data to file.
20/1-13	Set up a bugtracker to keep track of bugs and remaining work,
3/2-13	Created the web page for this project.
9/2-13	Released BeLT 0.2.0, with the following functionality; new way to filter data before output to screen, possible to send data to server using TLS, functioning installer for 32-bit version.
12/2-13	Released BeLT 0.2.1. A bit extra functionality, but mostly bug fixing.
24/2-13	Finished stress-test, using 22 computers.
25/2-13	Released BeLT 0.3.0. Ability to update, instead of reinstalling, bug fixes, performance optimization, and persistent storage of information.
27/2-13	Released BeLT 0.3.1. Small changes after meeting with employer.
5/3-13	Acquired certificates for our server and code signing certificate for our application.
10/3-13	Released BeLT 0.4.0. Changes and bug fixes, mostly thing under the hood of the application, very few visible changes.
11/3-13	Finalized the CSV format with employer
13/3-13	Released BeLT 0.4.1. Small changes after feedback from employer.
22/3-13	Released BeLT 0.4.2. Implemented automatic update functionality, including the server component.
5/4-13	Released BeLT 0.5.0. Major changes to the GUI and enhancements for how data is gathered. Most notable changes are settings dialog that stores the user settings, option to send a previously stored file to server.
9/4-13	Released BeLT 05.1. Small changes after feedback from the employer.
22/4-13	Released BeLT 0.6.0. Many changes to the GUI after feedback from the employer, this version has all the necessary functionality that has been developed under our meetings with the employer.
30/4-13	Released BeLT 0.7.0. Mostly bug-fixes, but some noticeable changes.
<i>Continuing on next page</i>	

<b>Date</b>	<b>Description</b>
10/5-13	Released BeLT 0.9.0. Released the latest version for feedback and any small changes that are needed before we release the last version.
15/5-13	Released BeLT 1.0.0. Final version.

Employer meeting for  
**BeLT** bachelor project

<b>Date &amp; time:</b>	14.01.2013 at 12:00
<b>Place:</b>	Kråkereiret
<b>Referent:</b>	Magnus Øverbø
<b>Participants:</b>	Magnus Øverbø, Lasse Johansen, Robin Stenvi, Patrick Bours, Soumik Mondal
<b>Absent:</b>	

<b>Agenda</b>
<b>Presentation of BeLT example-version</b>
<ol style="list-style-type: none"><li>1. UI Automation, key and mouse capture, etc.<ul style="list-style-type: none"><li>• Impressed about the current state</li></ul></li></ol>
<b>Discussion of the functions you want the program have</b>
<b>Format for storing the captures in (CSV or XML)</b>
<ol style="list-style-type: none"><li>1. CSV</li></ol>
<b>Storing of mousemovement</b>
<ol style="list-style-type: none"><li>1. Timestamps have an accuracy of +- 10ms</li><li>2. Can get better accuracy by using other methods<ul style="list-style-type: none"><li>• Don't use up resources with timestamping</li></ul></li></ol>
<b>Define "nonintrusive"</b>
<ol style="list-style-type: none"><li>1. Don't bug the user with messages</li><li>2. Minimize to system tray</li><li>3. The program should run in the background without user interaction</li></ol>
<b>Operational requirements</b>

## Agenda

1. Accuracy of mouse movement
  - GUI, possibility of deciding sampling rate and compression rate for mouse movement
2. Capacity of server/client
  - No worries about storage capacity
  - Store the login date and time at the start of log file
3. Transmission of data
  - Transmit the data at the end of the session
  - Build up chunks of data locally and then transmit at the end
  - Create new logfile at fixed intervals
4. Resource usage
5. Backwards compatibility on WinOS(Win7+, no need for more than XPSP3)
  - OS compatibility is not an issue, but shouldn't rely on special services
  - 64bit compatibility(Yes)
6. Other
  - Should be able to continue develop on the same system
  - Ability to add touchscreen in later times
  - Pause button for the application, with popup notification
  - Automatically detect password fields
  - Enumerate key/mouse events and show the log in GUI

### Suggestions to define the scope of the project

1. At least, Web browsers, Office and Skype
2. Collect, Store, Transmit is needed
3. Retrieval is not required, but would be great to have
4. Analysis is not required
5. Do not add touchscreen support
6. Retrieval of logs from server as CSV or text file
7. Noted and put into the requirement specification

### What is to be, or not to be, logged

## Agenda

1. HW information/status(Yes)
  - Store HW information/status on install of the system
2. Binary data(Screenshots, etc)(No)
3. Metadata(Filesize, MACtime)(No)
4. Contents of files(No)
5. External devices(HID and usb)
  - External human interfaces
  - Only keyboard and mouse
6. System status(Yes, noted)
7. GUI info(Yes, noted)
8. Physical switches, etc.(No)
  - No external switches
9. 2 entries per button(Yes)
10. 1 entry per button

### Security requirements

1. Server security
2. Client security
3. Security during transmission of data
4. Manipulation of temporary data
  - Client side security of temporary stored information is not necessary

### Privacy requirements

1. Anonymization of data
  - Anonymous storing, but identifiable across sessions
  - Stored on a secure server with no view access
  - Only grab the results of the analyzed log file
2. Removal of sensitiv information, we're currently able to detect password fields with UI Automation
  - Yes

### Prioritization of requirements

## Agenda

- Importance of metro-app compatibility(No need)
  - Importance of having a server/client based system(Needed, but can manage with only local system)
  - Importance of having a fully functioning and complete client only based system
  - Importance of testing, stresstesting(Important)
  - Importance of backwards compatibility of WinOS
  - Importance of removing sensitiv personal information(If possible remove password, otherwise not needed)
  - Importance of future development(Very)
  - Importance of having the files as CSV, as opposed to XML(Very important)
  - Importance of certification of software(Not a high priority, but a very nice feature)
- 1 Local version first
  - 2 Then storage on server and retrieval
  - 3 Analysis and touchscreen is future work

### Other items on the agenda

1. Most important to log the events, not analyze
2. Store everything in one huge file
3. Store mouse/key-event first then the occuring action

### For the next meeting

1. Prepare the software

# BeLT

## Behaviour Logging Tool

Magnus Øverbø(090832), Robin Stenvi(100232) og Lasse T. Johansen(090749)



Forprosjekt for bacheloroppgave  
IMT3912 - Bacheloroppgave IMT  
20 Studiepoeng  
Avdeling for informatikk og medieteknikk  
Høgskolen i Gjøvik, 2013



## Innhold

<b>Innhold</b> . . . . .	<b>ii</b>
<b>1 Bakgrunn, mål og rammer</b> . . . . .	<b>1</b>
1.1 Bakgrunn . . . . .	1
1.2 Prosjekt mål . . . . .	2
1.3 Rammer . . . . .	3
<b>2 Omfang</b> . . . . .	<b>3</b>
2.1 Oppgavebeskrivelse . . . . .	3
2.2 Avgrensning . . . . .	4
<b>3 Prosjektorganisering</b> . . . . .	<b>5</b>
3.1 Ansvarsforhold . . . . .	5
3.2 Rutiner og regler . . . . .	5
<b>4 Planlegging, oppfølging og rapportering</b> . . . . .	<b>7</b>
4.1 Hovedinndeling av prosjekt . . . . .	7
4.2 Plan for statusmøter og beslutningspunkter . . . . .	8
<b>5 Organisering av kvalitetssikring</b> . . . . .	<b>8</b>
5.1 Kvalitetssikring . . . . .	8
5.2 Risikoanalyse . . . . .	9
<b>6 Plan for gjennomføring</b> . . . . .	<b>11</b>
<b>Bibliografi</b> . . . . .	<b>14</b>

# 1 Bakgrunn, mål og rammer

## 1.1 Bakgrunn

### Prosjektets bakgrunn

Soumik Mondal jobber med å gjenkjenne brukermønstre for å autentisere databrukere for NISLab (Norwegian Information Security laboratory).

I denne sammenheng er det nødvendig å samle informasjon om hvordan brukere interakterer med datamaskinen, som tastetrykk og musebevegelser, men også hvordan brukeren jobber med forskjellige applikasjoner, da det ofte er mange forskjellige måter å gjøre samme oppgave på. For å lagre et dokument for eksempel, kan man trykke "ctrl+S", man kan gå opp til fil-menyen og trykke på lagre, eller man kan bruke knappen som ligger på knappelinjen. Disse tre funksjonene har akkurat samme utfall, men forskjellige personer har forskjellige preferanser.

For at analysen skal bli generell, trengs det mange testbrukere, så det er ønskelig med en applikasjon som kan distribueres til mange datamaskiner og samle dataene på en sentral server.

### Våres bakgrunn

Alle i gruppen studerer bachelor i informasjonssikkerhet ved Høgskolen i Gjøvik. Ingen av gruppe-medlemmene har noen spesiell erfaring i å programmere opp mot Windows sitt API, men alle har programmert mye i C++, gjennom fagene, "Grunnleggende programmering", "Objektorientert programmering" og "Algoritmer" på skolen. Andre fag som kommer til å være en nyttig er:

**Operativsystemer:** Vi trenger å vite om sikkerhetsmekanismene i Windows og hvordan Windows er strukturert og designet.

**Programvaresikkerhet:** Viktig at applikasjonen er utviklet med sikkerhet i tankene fra starten av. Bygger også videre på sikkerhetsmekanismene i Windows.

**Informasjonsstrukturer og databasemodellering:** Vi trenger å finne en effektiv og god løsning på hvordan dataene skal lagres.

**Systemutvikling:** To av gruppe-medlemmene har hatt systemutvikling, som vil være nyttig i å utvikle et produkt som er det kunden ser ute etter og generell prosjektgjennomføring.

**Systemadministrasjon:** Ett av gruppe-medlemmene har hatt systemadministrasjon som vil komme god med når vi skal sette opp serveren gjøre applikasjonen klart for distribusjon.

**Ethical hacking and penetration testing:** Dette vil hjelpe oss med å vite hvilke trusler som finnes mot applikasjonen vår og kunne teste applikasjonen mot disse truslene.

**Risikostyring:** Dette vil hjelpe oss med å analysere risikoen og utarbeide tiltak til hvordan den burde håndteres.

## Relatert arbeid

Det er blitt gjort en del arbeid for å fange musebevegelser[1, 2, 3], i de fleste er målet å finne ut hvordan brukeren jobber opp mot en applikasjon for å forbedre brukeropplevelsen. Disse er alle rettet mot web-applikasjoner og fokuserer på å visualisere hvordan brukeren interakterer med en web-applikasjon.

RUI[4] er et program laget i C# for Windows og Carbon Framework til Mac OS X. Programmet lagrer kun tastetrykk og musebevegelser, lagrer altså ingen interaksjon med applikasjoner.

En annen artikkel[5] prøver å lære brukerens interesser ved å observere oppførselen. Lagrer få data og er dermed ikke så relevant for oss.

EAGER[6] er et program som lagrer oppgaver som brukeren foretar seg og programmerer repetitive oppgaver. Eldre artikkel som har lite av verdi for oss.

AppMonitor[7] er en applikasjon som lagrer brukeraksjoner for å finne ut hvordan brukeren interakterer med datamaskinen. Den bruker Windows hooking for å fange musebevegelser og tastetrykk og den bruker Microsoft Active Accessibility for å fange hvordan brukeren interakterer med applikasjoner. AppMonitor støtter kun Adobe Reader 7 og Word 2003. Dette er et arbeid som er ganske likt det vi prøver å få til og en nyttig ressurs.

## 1.2 Prosjektmål

### Effektmål

Det forventes at det ferdige programmet skal forbedre følgende:

- Forenkle prosessen med å samle inn brukerinformatjon<sup>1</sup>.
- Samle inn mer brukerinformatjon, både fra flere brukere og fra et større tidsperspektiv.
- Samle inn mer nøyaktig informasjon om brukere og plassere det inn i den riktige kontekst, slik at analysen blir lettere og mer effektiv.

### Resultatmål

Applikasjonen skal leveres innen 15. Mai 2013 og skal ha følgende funksjonalitet:

- Fange tastetrykk.
- Fange musebevegelser.
- Lagre hvordan brukeren bruker applikasjoner. Nyttig ting er, om det brukes hotkeys, menyer, knapper osv.
- Sende og lagre informasjonen til en sentral server.

Dette skal gjøres på en måte som er sikkert og ikke-intrusiv for brukeren.

Rapporten skal bidra med følgende:

- Utarbeidelse av etiske problemstillinger, knyttet til slik programvare og hvordan dette bør håndteres.
- Utarbeidelse av en risikovurdering ved distribuering av programvaren og hvordan den kan

---

<sup>1</sup>Brukerinformasjon her regnes som alt det brukeren gjør for å interaktere med programmer

minimeres. Dette gjelder vurdering av klient-applikasjonen, server-applikasjonen og oversending av sensitive data.

### 1.3 Rammer

Gruppen må forholde seg til følgende tidsfrister:

27/1-13 Innlevering av prosjektplan og prosjektavtale

10/2-13 Etablering av nettside

15/5-13 Innlevering av Prosjektrapport

29/5-13 Innlevering av plakat til utstilling

4-6/6-13 Presentasjon av prosjektet

Kontaktpersoner er:

- Hanno Langweg – Veileder
- Soumik Mondal – Oppdragsgiver
- Patrick Bours – Interessepart

## 2 Omfang

### 2.1 Oppgavebeskrivelse

Oppgaven går ut på å lage en applikasjon som fanger interaksjonen brukeren har med data-maskinen. Informasjonen skal oversendes og lagres på en sentral server. Deretter skal det være mulig å eksportere dataene på et lett forståelig format.

Gruppen må undersøke muligheten for å fange interaksjon med alle applikasjoner ved en metode, eller en metode for å fange interaksjon med helt nye applikasjoner. Dette vil sørge for at applikasjonen er universell og man trenger ikke å oppdatere applikasjonen når nye versjoner av programvare kommer. Hvis dette ikke er mulig skal applikasjonen minst håndtere:

- Microsoft Office
- En eller flere PDF-lesere
- En eller flere nettlesere
- Skype

Applikasjonen skal være designet slik at det er mulig å utvide denne listen på et senere tidspunkt, uten alt for mye problemer.

Oppgaven kan deles inn i seks deler:

**Fange tastetrykk:** Her må vi først utarbeide hva som er den beste mulige måten å gjøre dette på. Viktig at måten er nøyaktig, tid må være nøyaktig opp til ett millisekund. Det må være effektivt og kompatibelt på flest mulige Windows plattformer.

**Fange musebevegelser:** Her har vi de samme oppgavene som ovenfor, men vi må også finne et bra format, å lagre musebevegelser på, da dette kan bli mye data etter kort tid. I programmet RUI[4] som ble nevnt tidligere, ble det lagret omtrent 22 KB/min i musebevegelser. Dette er ikke noe stort problem, men kan minkes.

**Fange interaksjon med applikasjoner:** Hvordan arbeider brukeren med applikasjonen, menyer, hurtigtaster osv. For at dette skal være universelt for alle applikasjoner, må det detekteres når to forskjellige brukerinteraksjoner gir likt resultat.

**Lagring av informasjon:** Det må først utvikles et format for lagring av dataene, deretter må det lages en løsning for sending og lagring av disse dataene på en sentral lokasjon. Denne løsningen må bevare konfidensialitet, integritet og håndtere personvern hensyn.

**Eksporere informasjon:** Det må utarbeides et fornuftig format å presentere dataene på, dette kan for eksempel være CSV, eller XML format, eventuelt flere muligheter. Formatet skal være leselig uten bruk av programmer.

**Utarbeide risikoanalyse for bruk av applikasjonen:** Hvilke risikoer er viktige hvis denne applikasjonen distribueres til tusen brukere og alle kobler mot en sentral server. Hvordan kan vi håndtere og minimere denne risikoen.

Et annet viktig område som det må sees nærmere på er tids granularitet. Det er viktig at når brukeren gjør noe for eksempel når et tastetrykk, så blir tiden tatt, med nøyaktighet ned mot ett millisekund. Klokkene på datamaskiner er ikke lagd for å være helt nøyaktige, og hvis du henter en verdi som måles i millisekunder, betyr ikke dette at tiden oppdateres hvert millisekund. Her kan nøyaktigheten variere stort mellom forskjellige systemer.

Prosjektet skal også utarbeide personvern hensyn ved bruk av applikasjonen og utarbeide hvilke risikoer som er relevante ved distribuering av applikasjonen og hvordan disse kan håndteres.

Selve applikasjonen kan deles i to deler, en er klient-delen som samler inn data. Den andre er en servertjeneste som tar imot informasjonen og lagrer dataene på et fornuftig format, for eksempel en komma separert liste. Viktig at denne tjeneren er sikker, da den skal muligens kjøre på en sensitiv server og den tar i mot sensitiv data.

I prosjektet må vi også se på sikkerhetsrisikoer, servertjenesten behandler sensitive data og må være programmert sikker. Klient-applikasjonen kommuniserer på nettet, vi må sørge for at applikasjonen vi lager ikke introduserer brukeren for mer risiko.

Prosjektet må også ta for seg personvern hensynet, både i følge personopplysningsloven og i forhold til etiske rammeverk. Personvernet bør i all hovedsak løses ved tekniske løsninger, data lagret på serveren bør ikke kunne knyttes til en person så vidt dette er mulig.

## 2.2 Avgrensning

Applikasjonen som utvikles vil kun ta for seg Windows operativsystemet. Det er ønsket bakoverstøtte så langt som mulig, men grunnet teknologivalg, kommer applikasjonen kun til å være støttet på Windows XP SP3 og nyere. Dette kan også bli begrenset på bakgrunn av fremtidige valg.

Applikasjonen skal ikke analysere dataene, men kunne samle de inn.

Det er ikke nødvendig å lage støtte til touchscreen, våres applikasjon skal kun støtte mus og tastatur.

## 3 Prosjektorganisering

### 3.1 Ansvarsforhold

Prosjekt/gruppeleder:	Robin Stenvi
Webmaster, systemansvarlig:	Lasse Tjensvold Johansen
Referent, utstyr og dokumentansvarlig:	Magnus Øverbø

**Prosjekt/gruppeleder:** Har ansvaret for å delegere arbeidsoppgaver til resten av gruppen, samt sørge for at tidsfristene som blir overholdt. I tillegg har prosjektlederen ansvaret for å lede møtene som holdes gjennom prosjektet og holde oversikten over prosjektets gang.

**Webmaster, systemansvarlig:** Har hovedansvaret for å opprettholde prosjektets weblogg, systemtjenester og GANTT-skjema vi benytter. Ansvar for oppdatering av weblogg faller i tillegg på resten av gruppen.

**Referent, utstyr/dokumentansvarlig:** Referenten har ansvaret for å ta referat av møtene, både internt i gruppen og eksternt. Dokumentansvarlig har ansvaret for å holde oppsyn over dokumentasjon, og repoet som benyttes under prosjektet. Utstyrsansvarlig har ansvaret for å anskaffe det utstyret som gruppen trenger for å utføre sitt arbeid.

### 3.2 Rutiner og regler

Alle gruppemedlemmer plikter seg til å overholde de følgende reglene. Gjør man ikke dette vil det forekomme sanksjoner.

#### Generelle regler

- Alle gruppemedlemmer skal gjøre sitt ytterste for å lykkes med prosjektet og møte til samtlige møter og arbeidsøkter.
- Det skal iblant forekomme et sosialt samvær hvor man gjør noe annet enn prosjekt/skolearbeid.
- Felles inntekt/utgift splittes likt mellom gruppemedlemmene.

#### Utvikling

- Vi benytter HiGs SVN repo-løsning for å vedlikeholde vår kodebase og utviklingsprosess, <https://svn.hig.no/2014/belt/>.
- For kontinuerlig integrasjon vil vi benytte scriptene vi har laget.
- Feil som oppstår i repoet skal rettes øyeblikkelig av vedkommende eller andre i gruppen.
- Alle medlemmer arbeider hovedsaklig på separate oppgaver.
- En tildelt oppgave skal ikke gjøres av en annen, med mindre det er gyldig grunnlag for det.

- Det skal kun lastes opp kompilerbar kode/tekst til repoet.
- Hvis det er grunnlag kan man laste opp ikke-kjørbar kode etter opplysning om dette til resten av gruppen, samt gi en beskrivende kommentar i repoet.
- Alle opplastinger skal ha en ordentlig kommentar vedlagt.

## Standarder

- Bruk av usikre funksjonskall er ikke akseptabelt. Dette gjelder funksjonskallene som Microsoft har bannlyst her: <http://msdn.microsoft.com/en-us/library/bb288454.aspx>.
- All kode skal kommenteres, med mindre det er elementært.
- All kode skal kodes iht. Doxygen. <http://www.stack.nl/~dimitri/doxygen/index.html>

## Kommunikasjon

- Kommunikasjon forekommer normalt sett muntlig, men med muntlig resymé i en etterfølgende e-post.
- All kommunikasjon over e-post skal sendes via HiGs epost tjeneste
- Alle e-poster skal starte med "[BELT]"i emne-området for å identifisere e-poster vedrørende prosjektet.
- Kritisk informasjon skal gis i form av SMS eller oppringing, etterfulgt av en e-post.
- Alle medlemmer skal sjekke deres HiG e-post to ganger daglig iløpet av arbeidsdager og en gang iløpet av helgen.
- Websiden skal oppdateres med nye innlegg minst en gang i uken.

## Møte, loggføring, stemming og signering

- Møteinnkallelse til ekstra møter skal sendes minst tre arbeidsdager i forvei, med mindre det er ekstremt viktig med det evt. møtet.
- Referater distribueres med SVN repository, senest 24 timer etter møtet. Dette er referent ansvarlig for.
- Interne tidsfrister bestemmes fortløpende under prosjektets gang.
- Alle medlemmer skal holde sin egen loggbok over deres arbeid, minimum ett innlegg per uke.
- Stemming gjøres i felles samvær.
- Signering av dokumenter skal alltid gjøre i samarbeid med de andre gruppemedlemmene.
- Under spesielle omstendigheter kan signering/bestemmelser gjøres alene, men kun når det ikke er mulig å gjøre det i samarbeid med et medlem til.

### **Gyldig fravær og beskjedgivning.**

- Alle beskjeder skal gis så tidlig som mulig.
- Beskjeder om planlagt fravær må gis minst tre arbeidsdager før fraværet skal inntreffe
- Andre beskjeder må gis så tidlig som mulig, før et møte/tidsfrist/annet forekommer.
- Sykdom, ved mer enn en liten forkjølelse
- Forhåndsavtalt fravær, som er godtatt i fellesskap.
- Uventede hendelser(Kriser, familiekriser, helse/død)
- Andre grunner som i etterkant godtas i fellesskap.

### **Sanksjoner**

- Sanksjoner kan kun gis hvis det ikke foreligger en gyldig grunn.
- Unnværelse å levere inn arbeidsoppgaver innen fristen resulterer i en skriftlig advarsel via epost.
- Tre advarsler resulterer i et møte med veileder for å bestemme håndteringen av situasjonen, med mulighet for ekskludering

## **4 Planlegging, oppfølging og rapportering**

### **4.1 Hovedinndeling av prosjekt**

For bachelorprosjektet vil vi benytte en iterativ systemutviklingsprosess. Utviklingsmetodikken vil ha faste lengder, hvor hver runde vil ende i et ferdig utviklet produkt som blir distribuert til arbeidsgiver. Vi har valgt å sette av ti arbeidsdager til hver iterasjon, som igjen vil resultere i rundt ni distribusjoner av programvaren til arbeidsgiver.

Grunnen for å velge en iterativ prosess er at det gir oss en mulighet for å bygge programvaren i inkremitter, som til slutt ender i en fullverdig versjon 1.0 av programmet I tillegg muliggjør det for oss å benytte arbeidsgiver som testpersonell gjennom hele prosessen, slik at vi får tilbakemeldinger av arbeidsgiver etter hver distribusjon av programvaren.

Prosjektet er delt inn i to spor som går parallelt, spor nummer én er utviklingsprosjektet, spor to er utarbeidelsen av prosjektrapporten. Prosjektrapporten er basert rundt noen få hovedpunkter. Først kommer teorien bak vår prosjekt og hvordan vi har satt opp våre systemer. Deretter kommer teorien om vårt prosjekt, hva vi har gjort, oppdaget og utviklet. Til slutt kommer delen om problematikken rundt personvern, hvordan dette sett er i sammenheng med vårt prosjekt og sluttprodukt. Mye av oppgaven vår er nært knyttet opp i mot personvern siden vi loggfører brukerinformasjon, både sensitiv og generell data. Spesielt iht. at produktet vårt loggfører og lagrer brukerens tastetrykk og musebevegelser under en sesjon hos en sentral serverenhet.



Utviklingssporet er delt inn i sine respektive deler, hvor vi først utarbeider en skrivebords-applikasjon brukeren kan benytte på sitt eget system. Del to blir å utvikle server-programmet, samt etablere kommunikasjon mellom klient og server. Del tre baserer seg på å utvikle en driver for I/O av mus og tastatur, samt en egen metro-applikasjon for å teste mulighetene innenfor dette området.

For å sikre fremgang i prosessen har vi satt av to møter i uken internt i gruppen på mandag og torsdag. Disse møtene er for at gruppen skal kunne samles og utveksle idéer, status på oppgaver og videre utførelse av oppgaven.

I tillegg til interne gruppemøter to ganger i uken har vi avsatt ukentlige møter med veileder på fredag formiddag. Disse møtene er ment for å kunne komme med foreløpig status på oppgavene utført, og få en rask tilbakemelding på arbeidet utført. Disse møtene sikrer at vi holder oss innenfor rammene av oppgaven, samt holder en rød tråd rundt prosjektet.

Til slutt har vi møte med arbeidsgiver annenhver uke for å utlevere en ny versjon av programvaren, samt motta tilbakemelding på foregående distribusjon. Disse møtene øker sjansene for at arbeidsgivers behov til programvaren blir ivaretatt og at arbeidsgiver er tilfreds med det endelige produktet.

## 4.2 Plan for statusmøter og beslutningspunkter

- Periodiske gruppemøter skal holdes på mandag kl. 09:00 og torsdag kl. 14:00.
- Møter med veileder skal holdes på fredag kl. 11:30-12:00.
- Møter med arbeidsgiver holdes på mandager kl. 12:00.

### Beslutningstaking

- Beslutninger tas i hovedsak som avstemning i felles samvær mellom alle gruppe-medlemmene.
- Beslutninger og signering av dokumenter skal alltid gjøre i samvær med de andre gruppe-medlemmene.
- Under spesielle omstendigheter kan beslutninger/signering gjøres alene, men kun når det ikke er mulig å gjøre det i samarbeid med et medlem til.

## 5 Organisering av kvalitetssikring

### 5.1 Kvalitetssikring

For å sikre kvaliteten av vårt utviklingsprosjekt har vi implementert versjonskontroll, kontinuerlig integrasjon og statistisk analyse av programvaren. For å sikre samhandling benytter vi Høgskolen i Gjøviks versjonskontrollsystem, Subversion. Dette gir hele gruppen et felles lagringsområde hvor man kan hente ut og legge inn kode til prosjektet. For å hindre at feil i programvaren blir lagt til har vi implementert vår egen versjon av kontinuerlig integrasjon ved å lage egne skript

som tillater å laste opp kode kun når den kompilerer, den statiske analysen er godtatt og ingen konflikter med eksisterende kode oppstår.

Dermed sikrer vi at koden som ligger på serveren er kjørbart og sikker i størst mulig grad.

For å utføre statistisk analyse av kildekoden benytter vi oss av CPPCHECK som utfører statistisk analyse av kildekoden og returnerer de feil den finner. I tillegg benytter vi oss også av Microsoft Visual Studio 2012 sin egen kodeanalyseverktøy. Dette er for å ha to uavhengige kilder for feilsøking av koden. CPPCHECK vil kjøre hver gang man prøver å bygge koden, siden denne er relativt rask til å kjøre. Deretter vil vi benytte Visual Studios analyseverktøy før hver innlevering så vi har en ekstra analyse av programmet.

For å for å få ekstra sikkerhetsadvarsler ved bygging av programvaren har vi slått på Security Development Lifecycle Checks(SDL checks). Dette legger til en rekke anbefalte sjekker, bl.a ekstra funksjoner for generering av sikker kode. I tillegg vil den gi ekstra tilbakemeldinger med tanke på sikkerhetsadvarsler.

Vi har laget skript som integrerer subversion, statistisk analyse av kildedekode og kompilering i Microsoft Visual Studio. For kommentering av kode har vi valgt å benytte Doxygen. Siden dette gir oss muligheten til å eksportere kommentarer til  $\LaTeX$  og HTML har vi valgt dette for å enklere knytte informasjon fra kildekoden inn i rapporten.

For å øke sjansene for at arbeidsgiver blir fornøyd med produktet setter vi opp en "Bug Tracker" hvor testerene kan fortløpende legge inn meldinger om feil i produktet, slik at vi ikke går glipp av det på et møte og at feilen kan evt. løses med en gang vi har tid.

## 5.2 Risikoanalyse

Vi har gjennomført en forenklet risikoanalyse hvor vi har vektet faktorene med verdiene høy-middels-lav for å angi deres påvirkning på prosjektet og sluttproduktet. Faktorene vi har valgt er *sannsynligheten* for at trusselen vil inntreffe under prosjektet og hvilken *konsekvens* det vil få for sluttproduktet om den inntreffer. Summen av sannsynligheten og konsekvensen er *risikoen* trusselen utgjør for prosjektet.

Trussel	Sannsynlighet	Konsekvens	Risiko
Tap av medlem	lav	høy	middels
<b>Beskrivelse</b>	Det kan skje uforutsette hendelser som medfører at et eller flere gruppemedlemmer ikke kan utføre sine arbeidsoppgaver; dette kan for eksempel være død i nær familie, force majeure, at gruppemedlemmet blir skadet og langtidssykemeldt eller at medlemmet dør		
<b>Tiltak</b>	Vi har iverksatt tiltak for å holde samholdet ved like. Vi vil holde sosiale sammenkomster utenom prosjektet, men fysisk skade er dessverre ikke mulig å iverksette tiltak imot.		
Ikke godkjent sluttprodukt av arbeidsgiver	lav	middels	lav
<b>Beskrivelse</b>	Det er en sannsynlighet for at det endelige produktet ikke holder mål iht. arbeidsgivers krav, dette kan for eksempel være at produktet er ustabil, ikke brukervennlig eller uferdig		
Fortsetter på neste side			

## Behaviour Logging Tool

Trussel	Sannsynlighet	Konsekvens	Risiko
<b>Tiltak</b>	<i>Vi har involvert arbeidsgiver tidlig i prosessen og vi har valgt å benytte korte iterasjoner i utviklingen. Arbeidsgiver har allerede tidlig vist interesse for å hjelpe til med testing av produktet. Vi vil sette opp en bugtracker hvor det kan rapporteres feil når som helst til en sentralisert enhet som vi følger med på. Dette senker sannsynligheten for at sluttproduktet ikke blir godkjent.</i>		
<i>Halvferdig klient/server-arkitektur</i>	middels	middels	middels
<b>Beskrivelse</b>	<i>Det er en sjanse for at vi møter uforutsette hindringer som gjør at vi ikke får nok tid til å ferdigstille transmisjonsbiten av programmet, dette er en risiko vi må ta inn over oss i arbeidet vårt.</i>		
<b>Tiltak</b>	<i>Siden vi jobber etter en inkrementell utviklingsmodell vil vi uansett ha et kjørbart program etter hver iterasjon. Dermed vil det vi har utviklet til det stadiet være i orden og kjørbart med et utvalg av den ønskede funksjonaliteten på plass. Derfor er ikke dette et risikotap uansett hva som skjer.</i>		
<i>Utdatert sluttprodukt</i>	lav	middels	lav
<b>Beskrivelse</b>	<i>Det kan hende at de teknologivalgene vi baserer oss på viser seg å være utdatert når vi kommer til innleveringsfasen, sannsynligheten er lav på grunnlag av teknologiens art.</i>		
<b>Tiltak</b>	<i>I våre teknologivalg vil vi først gjennomføre et selvstudie, for å så prate med ressurspersoner på høgsolen hvis det er noe vi er i tvil om. Det ligger masse dokumentasjon ute på internett, og høgsolbiblioteket har et godt utvalg av bøker som omhandler Windows-programmering og -arkitektur. Ved å være bevisst på hvordan vi behandler informasjonen vi innhenter, minsker vi sannsynligheten for at informasjonen er enten a) feil eller b) utdatert.</i>		
<i>Endring av lovverk, krav, standarder</i>	lav	lav	lav
<b>Beskrivelse</b>	<i>Sannsynligheten for at et lovverk endres er liten, men at det forekommer nye krav, enten i lovverk eller standarder, som hindrer produktet i å bli brukt til sitt formål kan forekomme.</i>		
<b>Tiltak</b>	<i>Siden programvaren kun vil benyttes i et forskningsprosjekt vil programvaren kun benyttes i et kontrollert miljø hvor det opplyses om hva programvaren gjør, og loggfører. Dermed vil ansvaret falle på sluttbrukeren.</i>		
<i>Uoppdaget sikkerhetshull</i>	middels	høy	middels
<b>Beskrivelse</b>	<i>Siden sikkerhetshull er vanskelig å oppdage er det en sjanse for at det foreligger programmeringsfeil, enten fra oss eller fra tredjeparts systemer. Disse feilene kan resultere i sikkerhetshull.</i>		
<b>Tiltak</b>	<i>Ved å implementere statistisk analyse av kode, kontinuerlig integrasjon og kodegjennomgang vil sannsynligheten for dette reduseres.</i>		
<i>Liknende produkt slippes før vårt</i>	lav	lav	lav

Fortsetter på neste side

Trussel	Sannsynlighet	Konsekvens	Risiko
<b>Beskrivelse</b>	<i>Det er en sannsynlighet for at det lages tilsvarende programvare et annet sted i landet/verden som vi ikke er klar over.</i>		
<b>Tiltak</b>	<i>Siden vi har dette som et fastsatt bachelorprosjekt vil ikke dette påvirke oss.</i>		
<i>Misforståelser</i>	middels	middels	middels
<b>Beskrivelse</b>	<i>Innad i gruppen, arbeidsgiver, veileder, etc.</i>		
<b>Tiltak</b>	<i>Ved å ha hyppige møter åpner vi for dialog hvor vi kan luften idéer og fortere se og ta tak i misforståelser.</i>		
<i>Uteværende utstyr</i>	middels	middels	middels
<b>Beskrivelse</b>	<i>At utstyr vi trenger for å fullføre oppgaven på en tilstrekkelig måte ikke blir anskaffet.</i>		
<b>Tiltak</b>	<i>Vi baserer oppgaven minst mulig på utstyr vi må anskaffe fra en tredjepart, evt. utstyr vi skaffer skal ikke være kritisk for utviklingen av prosjektet.</i>		

## 6 Plan for gjennomføring

Dette kapitlet omhandler ressurser og tidsplaner. Her er en liste over personene, systemene og softwaren vi kommer til å bruke mest:

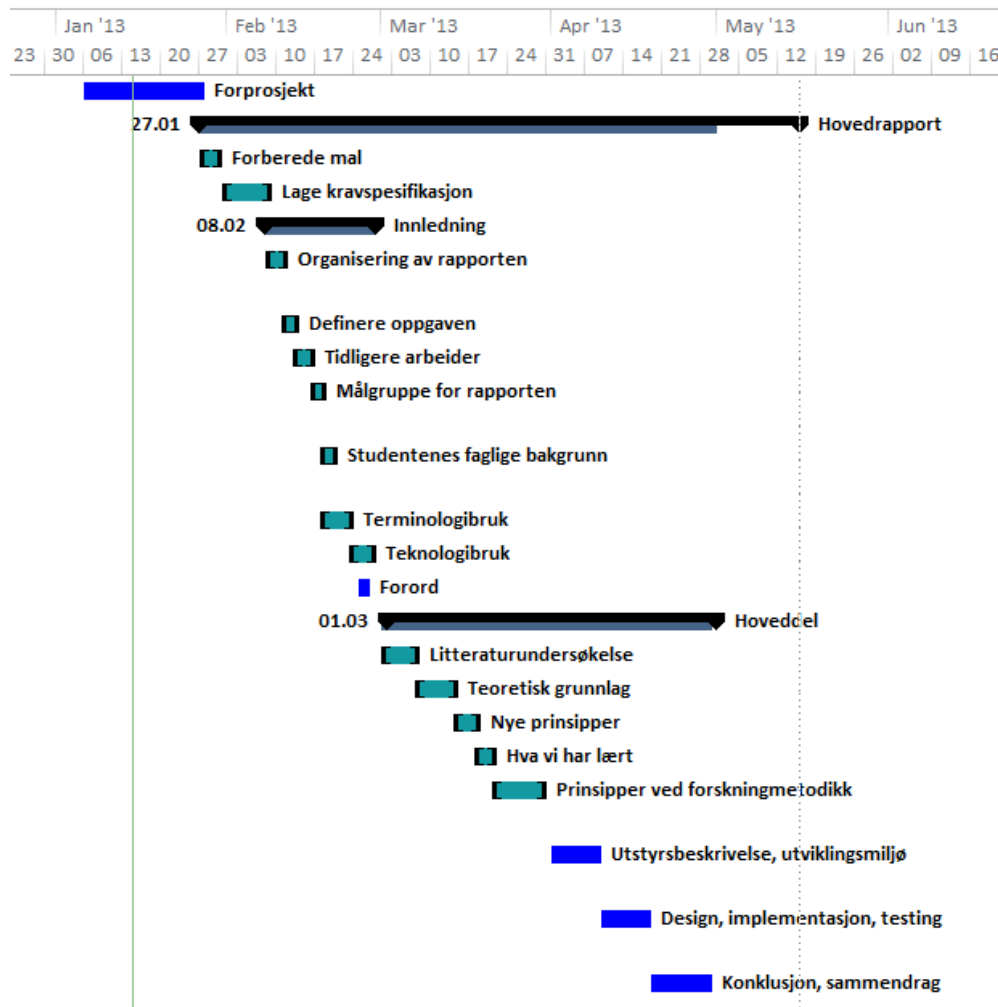
Ressurskategori	Ressurs	Beskrivelse
Personell	Robin	Gruppeleder
Personell	Magnus	Gruppemedlem
Personell	Lasse	Gruppemedlem
Personell	Soumik Mondal	Testperson, arbeidsgiver
Personell	Patrick Bours	Testperson, interessert part
Personell	Hanno Langweg	Prosess-veileder
Programvare	Visual Studio Professional/Premium 2012	For utvikling av kode
Programvare	Windows 8/7/XP	For testing og utvikling av programmet
Programvare	C++-check	For statisk analyse av kode
Programvare	SDK/WDK	Utviklingstillegg for Windows
Programvare	Psake	Powershell modul for forenkling av kompileringoppgaver
Programvare	Visual-svn	Tillegg til visual studio for håndtering av repositories
Programvare	Microsoft Project 2013	For utvikling av framdriftsplaner
System	Server HiG	Kontaktperson Erik Hjelmås
System	Subversion HiG	System for versjonskontroll av kode
System	Doxygen	System for å samle kommentarer fra kildekoden

Vi har delt vår plan for gjennomføring i to deler, en for utvikling av verktøyet og en for rapportskiving. Vi representerer tankemåten vår med to gantt-skjemaer i henhold til dette. Legg merke til at vi ikke har planlagt utviklingen lenger enn til mars, dette er fordi at det er vanskelig å forutsi hvordan utviklingen vil gå. Dette gir oss muligheten til å forskyve ugjorte oppgaver videre til neste iterasjon, og for eksempel legge til en iterasjon på slutten.

## Gantt



Figur 1: Plan for utvikling



Figur 2: Plan for rapportskrivning

## Bibliografi

- [1] Arroyo, E., Selker, T., & Wiy, W. 2006. Usability tool for analysis of web designs using mouse tracks. *CHI EA '06 CHI '06 Extended Abstracts on Human Factors in Computing Systems*, 484–489.
- [2] Atterer, R., Wnuk, M., & Schmidt, A. 2006. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. *WWW '06 Proceedings of the 15th international conference on World Wide Web*, 203–212.
- [3] Mueller, F. & Lockerd, A. 2001. Cheese: tracking mouse movement activity on websites, a tool for user modeling. *CHI EA '01 CHI '01 Extended Abstracts on Human Factors in Computing Systems*, 279–280.
- [4] Kukreja, U., Stevenson, W. E., & Ritter, F. November 2006. RUI: Recording user input from interfaces under windows and Mac OS X. *Behavior Research Methods*, 38, 656–659.
- [5] Goecks, J. & Shavlik, J. 2000. Learning users' interests by unobtrusively observing their normal behavior. *IUI '00 Proceedings of the 5th international conference on Intelligent user interfaces*, 129–132.
- [6] Cypher, A. 1991. EAGER: programming repetitive tasks by example. *CHI '91 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 33–39.
- [7] Alexander, J. & Cockburn, A. Mai 2008. AppMonitor: A tool for recording user actions in unmodified windows applications. *Behavior Research Methods*, 40, 413–421.