

Aki Riisiö

# Windows Powershell Monitoring System

---

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

8.5.2017

Author Title	Aki Riisö Windows Powershell Monitoring System
Number of Pages Date	45 pages + 8 appendices 8 May 2017
Degree	Bachelor of Engineering
Degree Programme	Information technology
Specialisation option	Data networks
Instructor	Markku Nuutinen, Principal Lecturer
<p>Icinga is a free, open source monitoring system used worldwide. Companies such as Audi, Debian and McGill are relying on the extremely efficient services Icinga provides.</p> <p>To smaller companies e.g Canter Oy, monitoring services and servers are crucial for supplying customers with efficient support. Monitoring improves customer service by minimizing the reaction time if an issue emerges, whether a service has gone down, CPU usage is too high or backups are missing.</p> <p>While Icinga is free, it requires actions from corporate internet service providers (ISPs) to open connections to the customers' servers. The IP traffic between servers needs to be enabled and certain ports need to be opened.</p> <p>Every time a new customer is installed to Icinga, this forces Canter to send a support request to their ISP to make the desired changes and this leads to extra costs. This is time consuming and often the needed configurations cannot be done either due to VPN connections or security issues.</p> <p>The purpose of this thesis was to provide an alternate solution for service and server monitoring. A completely new monitoring system for Windows servers was created requiring no actions from ISPs. No IP traffic nor port modifications are needed. This also fixes the problem Icinga has had with VPN connections, since this new product sends the data from the servers directly to a cloud. It does not matter if the server is behind multiple firewalls or accessible only via VPN, only access to the internet is required.</p>	
Keywords	Canter, monitoring, Icinga, server

Tekijä Otsikko	Aki Riisiö Windows Powershell -valvontajärjestelmä
Sivumäärä Aika	45 sivua + 8 liitettä 8.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja	Yliopettaja Markku Nuutinen
<p>Insinööriyön tarkoitus oli toteuttaa vaihtoehtoinen ratkaisu palveluiden ja palvelimien valvonnalle. Koska kohdeyrityksen nykyinen valvontatyökalu, Icinga, riippuu vahvasti palveluntarjoajien toimenpiteistä, mm. portti- ja yhteysavauksista, aiheutuu tästä tarpeettomia kustannuksia aina, kun uusi asiakas lisätään valvontajärjestelmään.</p> <p>Työssä toteutettiin Powershellillä uudentyypinen valvontatyökalu, jossa data kerättiin valvottavalta palvelimelta ja lähetettiin Dropboxin kautta Linux-palvelimelle tietokantaan. Datat esittämistä varten laadittiin yksinkertainen verkkosivusto, jonka avulla tiedot pystyttiin esittämään joko tauluina tai graafisesti.</p> <p>Työssä luotu järjestelmä vastasi monin paikoin odotuksia ja toimi virheettömästi koko testijakson ajan (3 kk). Koska data lähetettiin pilven välityksellä tietokantapalvelimelle, ei palveluntarjoajaa tarvittu yhteyksien muodostamiseen ja näin valvontajärjestelmä osoittautui erittäin kustannustehokkaaksi. Tämä tarkoitti myös sitä, että palvelimet, joita ei tietoturvasyistä tai VPN:n (Virtual Private Network) takia pystytty valvomaan, saatiin kytkettyä järjestelmään.</p> <p>Pienellä jatkokehityksellä järjestelmästä saisi entistä tehokkaamman ja eheän. Uusien asiakkaiden asennuksien automatisointi säästäisi yhä enemmän aikaa, ja järjestelmän optimointi tietokannan ja verkkosivuston osalta parantaisi huomattavasti käytettävyyttä.</p>	
Avainsanat	Canter, valvonta, Icinga, palvelin, Powershell

## Table of contents

### Abbreviations

1	Introduction	1
1.1	Canter Oy	1
1.2	Icinga	2
1.2.1	Icinga Core and Icinga Web	2
1.2.2	Icinga Configuration	3
1.2.3	Pros and Cons	5
2	Environment	6
2.1	Host Server	6
2.1.1	Ubuntu Server 14.01	8
2.2	Client Server	8
3	Powershell	9
3.1	Scripts	11
3.1.1	Properties	11
3.1.2	Disk Space	12
3.1.3	Connections	14
3.1.4	Processes	16
3.1.5	System Time	17
3.1.6	Backups	18
3.1.7	Integrations	20
3.1.8	Jboss	22
3.1.9	Changed Processes	23
4	Configuration of Environments	25
4.1	Client Server	25
4.2	Ubuntu Server	27
4.2.1	Dropbox Client	28
4.2.2	Installing LAMP	29
4.2.3	Database	31
4.2.4	Generating Tables	33
4.2.5	Automating Data Flow	34
5	Website	38



5.1	Data Tables Page	38
5.2	JPGraph	41
6	Discussion and Conclusions	44
6.1	Further Development	45
	Sources	47
Appendixes		
	Appendix 1. Powershell scripts	
	Appendix 2. Script for creating the database tables	
	Appendix 3. Script for importing the CSV files	
	Appendix 4. Source code for the index page	
	Appendix 5. Source code for data tables page	
	Appendix 6. Source code for generating the graph	
	Appendix 7. styles.css file	

## Abbreviations

PIM            Product information management

ERP            Enterprise Resource Planning

ISP            Internet Service Provider

VPN            Virtual Private Network

CSV            Comma Separated Value

PHP            Personal Home Page or PHP: Hypertext Preprocessor

## 1 Introduction

To properly understand the purpose of the present study and why it was designed in this specific way, one needs to describe what Canter Oy, Windows Powershell, Nagios and Icinga are. Since Icinga is designed mostly for monitoring Linux machines, a third-party agent called NSClient has been created. This is an agent running on a Windows server which translates the commands from Nagios plugins to work with Windows machines.

### 1.1 Canter Oy

Canter Oy is a small software company focused on product information management. With 13 employees and a last year's revenue of 1.1 million euros Canter is the leader in the product information management branch in the domestic market.

The main product of Canter Oy is Adeona PIM. It is software for advanced data management. Adeona PIM runs on a server as a java process on the JBoss application server. This service has a local database connection to which the data is integrated, in most cases, from customers' ERP. Users can install the Adeona PIM client to their own computer which can be used to access the service running on the server. From Adeona PIM, the product data information can be distributed to channels specified by the customer. These channels can be for example web shop for consumers or businesses, online catalogues for sharing information inside the company or automatic publishing service.

Like any other service on the market Adeona has its issues. Whether it is a memory issue with the Java process, limited storage space on the server, connection issues between the server and a host or an error during data integrations, these will cause errors in the application. While it is nearly impossible to prevent these errors from happening, they can at least to some level be predicted. This is where monitoring comes in. To provide desirable customer service per the SLA levels, employees must have detailed information from the server constantly. If a process is not starting or data integration has not been able to finish properly, system managers must have the information before the customer contacts the support system or even better, before the customer even notices the problem.

At the moment, Canter is using Icinga to monitor the servers and services. This is discussed in the next chapter.

## 1.2 Icinga

Icinga is a free, open-source monitoring system running on a Linux server. It uses multiple plugins provided by Nagios to monitor servers installed to a network. Icinga is very robust, reliable and extremely efficient and is used mainly among ISPs.

### 1.2.1 Icinga Core and Icinga Web

Icinga has two main components, Icinga Web and Icinga Core [1]. The Core component is responsible for managing the scheduled checks using plugins provided by Nagios. In this introduction check\_nrpe (NRPE = Nagios Remote Plugin Executor) plugin is used since it is communicating directly with the NSClient on a Windows machine. For example, if an administrator needs to check the disk space of a Windows server directly from the Linux shell, the first step needed is to connect to the Windows machine from the Linux server. After the connection has been completed, the admin can define which plugin is used. Since the disk space needs to be checked, the CheckDriveSize plugin should be used. Figure 1 shows an example command to check disk space of a remote Windows machine using the CheckDriveSize command.

```
[canteradmin@canter-web01 plugins]$ ./check_nrpe -H 10.187.0.101 -p 5666 -c CheckDriveSize -a ShowAll MinWarn=5G MinCrit=2G Drive=c:  
OK c:: Total: 59.678GB - Used: 23.602GB (40%) - Free: 36.076GB (60%)|'c: free'=36.07608GB;5;2;0;59.67773 'c: free %'=60%;8;3;0;100  
[canteradmin@canter-web01 plugins]$
```

Figure 1. Example command to check disk space of a remote Windows machine.

The command in the Figure 1 is executed from a Linux command line. It has the parameter H, which is the address of the host. In the configuration files this is set as \$HOSTADDRESS\$ and it receives the correct IP address from another configuration file, localhost.cfg.

Parameter P defines the port where the NRPE-server is running. By default, this is 5666. C is the command which is entered to the NRPE-server. MinWarn and MinCrit parameters are set to tell when Icinga should set the status of the service to warning or

critical. In this case, the limit of warning status is if the amount of free space on drive C: is below 5 gigabytes.

Icinga Core is responsible for these commands. At the moment these commands are configured to run every 15 minutes. The results are filtered and stored in the Core's built in database called ido2db and are then displayed by the Icinga Web. This interface can be accessed from the local network where Icinga is installed. In this case, it can be accessed directly using the IP-address of the Linux server. The basic interface of Icinga Web can be seen in Figure 2.

Host	Service	Status	Last Check	Duration	Attempts	Checks/Command
MS0-Mediabank	HTTP-check	OK	21-12-2015 15:05:44	23.28s 40m 15s	1/5	HTTP OK: HTTP/1.1 200 OK - 8603 bytes in 0.304 second response time
ETRA-Adena	Adena-check	OK	21-12-2015 15:05:07	366.20s 24m 53s	1/4	OK: c: 52.4G
ETRA-Adena	Adena-check	OK	21-12-2015 14:58:24	366.20s 24m 53s	1/4	AMMS_ETRA_FULL_12202015_220000 BAK
ETRA-Adena	Adena-check	OK	21-12-2015 15:00:30	366.20s 24m 53s	1/4	AMMS_HESTEPHANE_FULL_12202015_220026 BAK
ETRA-Adena	Adena-check	OK	21-12-2015 15:08:48	366.20s 24m 49s	1/4	AMMS_PJEMETIC_FULL_12202015_220027 BAK
ETRA-Adena	Adena-check	OK	21-12-2015 15:01:29	336.0s 5m 36s	1/4	AMMS_TIME_FULL_12202015_220029 BAK
ETRA-Adena	Adena-check	OK	21-12-2015 14:59:10	1232.0s 40m 47s	1/5	OK: P04 100% (2) in open
ETRA-Mediabank	HTTP-check	OK	21-12-2015 14:55:16	216.0s 3m 41s	1/5	HTTP OK: HTTP/1.1 200 OK - 3428 bytes in 0.003 second response time
ETRA-Toolbox	HTTP-check	OK	21-12-2015 14:59:33	366.18s 42m 50s	1/5	HTTP OK: HTTP/1.1 200 OK - 4069 bytes in 0.005 second response time
ETRA-eCatalog	HTTP-check	OK	21-12-2015 15:07:39	216.7s 3m 36s	1/5	HTTP OK: HTTP/1.1 200 OK - 41873 bytes in 0.137 second response time
BK0-Mediabank	HTTP-check	CRITICAL	03-09-2015 11:05:03	1090.0s 3m 54s	5/5	CRITICAL - Socket timeout after 10 seconds
BK0-SERV-AdenaPH2	CPU-check	PENDING	N/A	230.0s 3m 45s	1/4	Service is not scheduled to be checked.
BK0-SERV-AdenaPH2	check-drive-space-C	PENDING	N/A	230.0s 3m 45s	1/4	Service is not scheduled to be checked.
BK0-SERV-AdenaPH2	check-drive-space-E	PENDING	N/A	230.0s 3m 45s	1/4	Service is not scheduled to be checked.
BK0-Servtool	HTTP-check	CRITICAL	03-09-2015 11:00:22	1090.0s 3m 55s	5/5	CRITICAL - Socket timeout after 10 seconds
IKU-Mediabank	HTTP-check	OK	21-12-2015 15:05:56	66.17s 30m 1s	1/5	HTTP OK: HTTP/1.1 200 OK - 10543 bytes in 0.245 second response time
Marlin-Server	Check-File-Size-ADMTM	OK	21-12-2015 15:06:02	360.18s 5m 5s	1/5	AMMS_IDOC_HOODEB000_CATT000002015121402020 001
Marlin-Server	Check-File-Count	OK	21-12-2015 15:00:19	1566.0s 25m 5s	1/5	5 Total files
Marlin-Server	Disk-Space-Check-C	OK	21-12-2015 15:00:25	366.18s 5m 6s	1/5	OK: c: 33.1G
Marlin-Server	Disk-Space-Check-D	OK	21-12-2015 14:54:42	44.2h 14m 15s	1/5	OK: d: 95.6G
Marlin-b2b-vestshop	HTTP-check	OK	21-12-2015 15:06:48	1180.18s 5m 39s	1/5	HTTP OK: HTTP/1.1 200 OK - 2212 bytes in 0.137 second response time
Marlin-b2b-vestshop	HTTP-check	OK	21-12-2015 15:01:05	1180.18s 5m 39s	1/5	HTTP OK: HTTP/1.1 200 OK - 18222 bytes in 0.102 second response time
Marlin-Mediabank	HTTP-check	OK	21-12-2015 14:57:11	36.18s 55m 40s	1/5	HTTP OK: HTTP/1.1 200 OK - 24878 bytes in 1.466 second response time
Medic-eCatalog	HTTP-check	OK	21-12-2015 15:01:29	366.20s 25m 42s	1/5	HTTP OK: HTTP/1.1 200 OK - 23078 bytes in 0.372 second response time
Hestephane-eCatalog	HTTP-check	OK	21-12-2015 15:07:34	78.9s 40m 22s	1/5	HTTP OK: HTTP/1.1 200 OK - 31954 bytes in 0.332 second response time
Pajama-eCatalog	HTTP-check	OK	21-12-2015 15:01:51	78.9s 37m 4s	1/5	HTTP OK: HTTP/1.1 200 OK - 42476 bytes in 0.295 second response time
Pajama-Mediabank	HTTP-check	OK	21-12-2015 14:54:58	66.18s 50m 59s	1/5	HTTP OK: HTTP/1.1 200 OK - 22278 bytes in 0.275 second response time
Talangen-eCatalog	HTTP-check	OK	21-12-2015 15:02:16	74.9s 30m 47s	1/5	HTTP OK: HTTP/1.1 200 OK - 18428 bytes in 0.310 second response time
Talangen-eREST	HTTP-check	OK	21-12-2015 15:01:20	66.19s 37m 37s	1/5	HTTP OK: HTTP/1.1 200 OK - 8563 bytes in 0.461 second response time

Figure 2. The interface of Icinga Web. Critical status is marked as red while OK status is marked as green.

List of installed hosts and services can be seen in Figure 2. Left pane is used for navigation and controlling the Icinga. User can stop, restart or shutdown the service.

### 1.2.2 Icinga Configuration

Installing and managing Icinga may be problematic for an inexperienced Linux user. The installation requires installing multiple components, such as Apache2, MySQL, IDOUTILS, PHP and many others. In addition, the configuration files for Icinga might be difficult to understand and when more and more customers are added to the monitoring service, the size of these configurations is greatly increased.

Icinga has two main configuration files: localhost.cfg and commands.cfg. Localhost.cfg includes all the hosts and their services and commands.cfg is used to define the syntax and commands for these services. At the moment, there are 14 customers installed to Icinga and the amount of services is 23. An average of 2 commands is configured to each service and the number of lines in the localhost and command configuration files are 1172 and 492.

Figures 3 and 4 below indicate what different parameters and configurations needs to be set before installing a new customer to Icinga.

```
define host{
    use                linux-server
    host_name          Tiivistekeskus-eCatalog
    alias              Tiivistekeskus-eCatalog
    address            10.187.0.100
    check_command      check_http_Tiivistekeskus-eCatalog
    max_check_attempts 5
    check_interval     5
    retry_interval     3
    check_period       24x7
    notification_interval 30
    notification_period workhours
    notification_options d,u,r,f
    notifications_enabled 1
    contacts           icingaadmin
}
```

Figure 3. Example of a host configuration in the localhost.cfg file.

```
define service{
    use                local-service
    host_name          ETOLA-canter-etra-sql01
    service_description check_SQL-Etra
    check_command      check_SQL-Etra
    check_interval     15
    check_period       24x7
    notification_interval 15
    notification_period 24x7
    notification_options w,u,c,r,f
    notifications_enabled 1
    contacts           icingaadmin
}
```

Figure 4. Example of a service configuration in the localhost.cfg file.

Figure 5 shows the basic method how commands are configured in the commands.cfg file. Distinctive names can be given to commands, even though they are using the available built-in Nagios plugins.

```

define command{
    command_name check_SQL-Tiivistekeskus
    command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -p 5666 -c CheckFiles -a
    -path=C:\\MSSQL_Backups\\Tiivistekeskus pattern="AAMS_TIKE_*" "filter=creation gt -25h" MinCrit=0
}

```

Figure 5. Example of a command to check if a file is newer than 25 hours in the commands.cfg file.

In the example above (Figure 5), the path variable is set to indicate the folder on the Windows machine. \$HOSTADDRESS\$ parameter lets Icinga know that the value from the host configuration file should be used. This can be seen in Figure 3. If the check\_command value in Figure 4 would be "check\_SQL-Tiivistekeskus" the \$HOSTADDRESS\$ parameter in Figure 5 would use the IP-address given to host "ETOLA-canter-etra-sql01".

### 1.2.3 Pros and Cons

One of Icinga's benefits is its reliability. Icinga has been in use for almost 3 years and it has had an error only once. The combination of low memory consumption and the robustness of the Red Hat Linux server ensures that Icinga performs well, assuming that all the necessary configurations are properly set.

The reason why a new way to monitor these servers was considered is not that Icinga would not perform well or that it is too difficult to configure, but that a way to filter out the third party needed to be found. As stated previously, Icinga uses NRPE to connect to Windows machines. The NRPE server on the Windows machine uses a specified port which Icinga can connect to. This means that if the port is closed by an ISP the connection cannot be established. Also, the traffic between networks by default is not allowed which means that every time a new customer or server is installed to Icinga the ISP needs to be contacted and a request to change the firewall rules to allow the traffic is needed. In addition, many customers have their own servers which means that they might have a different ISP. For example, Canter's ISP is MPY, while Rautakesko's servers are maintained by Tieto Oy. This means that even if MPY has enabled the traffic from Canter Oy network, the customer's ISP needs to be contacted as well to open the port and allow the connections to the server.

These requests take time and are often expensive and even if the connection has been opened by MPY, it does not mean that Tieto will open the connections on their side. This can be due to a security protocol or other policy issues.

There have also been issues installing a server to Icinga if the server has been behind VPN. Since Icinga requires direct Lan2Lan tunnels or routing configurations from the ISPs, direct connection over VPN cannot be achieved.

In this Powershell monitoring system, the required data could be sent without any problems even if the server was behind a VPN. While the requests to ISPs were mundane, they were no longer necessary since the data was sent via cloud service. This also removed the costs for installing a new customer to the Icinga monitoring system.

During the next chapters the environments on which the Powershell monitoring system and the example customer's server are running are introduced. Hardware, required software and necessary configurations are presented. The basic functionality of the Powershell and the scripts for data collection on the target server are examined as well as the process how the data is stored.

The end of the paper describes how the website for this system was built and how it can display the data stored in the database using free and open-source plugins. The issues and the development process are discussed in the last chapter.

## **2 Environment**

In this chapter, the hardware of the server which is monitored and the server on which the system is running are presented. These servers are called the host and the client server.

### **2.1 Host Server**

For development and testing purposes Canter has installed a server inside their office network called HADES. The server has i7-4770K CPU @ 3,50 GHz processor and 31,7 Gigabytes RAM. It has no operating system, but it has VMWare ESXi 5.5.0 (VMKernel Release Build 2068190) running on the hardware.



This setup enables a convenient installation and management of multiple virtual machines. ESXi is configured to use static IP address and therefore all the virtual machines can be controlled easily using the vSphere Client.

The core of the Powershell monitoring system is installed on Ubuntu Server 14.01. The Ubuntu Server has php5 libraries, mysql database, Apache2 web server and dropbox client installed. These are the key components of the monitoring system.

- PHP: database scripts, data to and from database to the website
- Mysql: database for the data from the client server
- Apache2: website to view the fetched data
- dropbox: client to transfer the data via cloud

Figure 6 displays the default interface in the vSphere Client and the complete list of installed virtual machines.

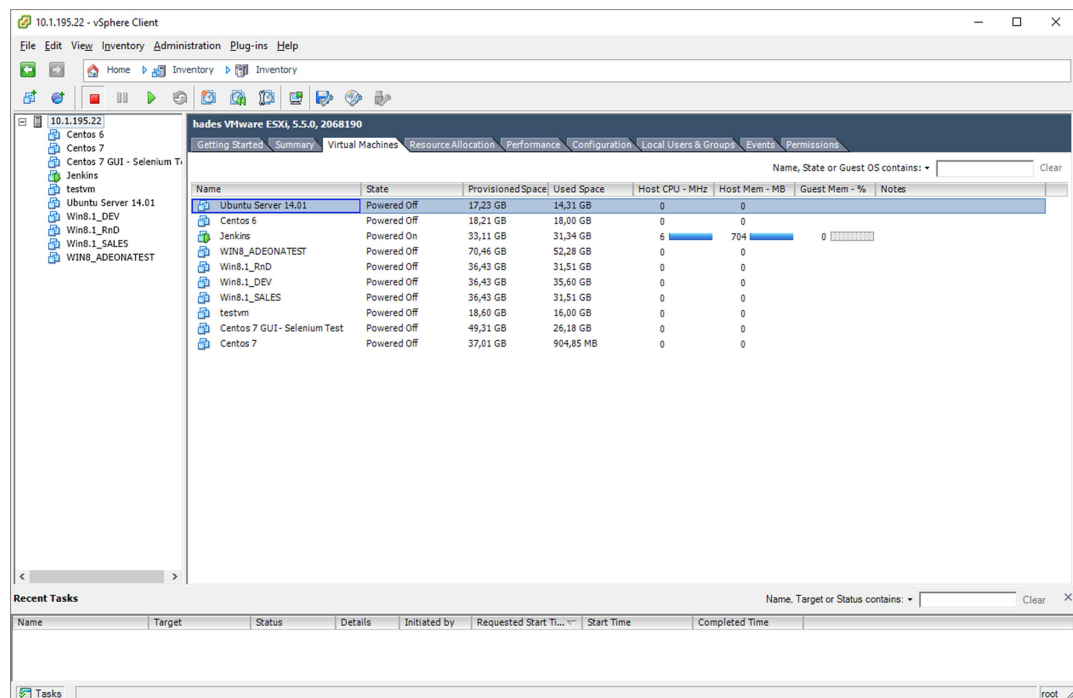


Figure 6. View of vSphere Client listing all virtual machines.

The virtual machine used in this project can be seen in Figure 6 on top of the list of the machines.

### 2.1.1 Ubuntu Server 14.01

The reason Ubuntu Server was chosen was that it was one of the few OS's where the installation of the dropbox client was fluent enough. Some of the other operating systems Canter uses, such as CentOS, were also considered but after spending multiple hours trying to find the proper configurations for the dropbox client to work with this OS, the idea was rejected.

Since HADES has more than enough RAM on it and it was concerning that Ubuntu Server would not be as stable as CentOS or RedHat, it was decided that 1 GB of RAM was allocated for the server. Installation of the Ubuntu Server was successful and no issues emerged during the configuration of the server.

There are a few key components in addition to the list introduced in the previous chapter. Cron is required to run certain scripts to handle incoming files and information security. Installation of the Dropbox client was not as easy as it was thought and it took multiple days to configure it properly. This issue was related to the latest version of the Dropbox and version degrade was needed.

## 2.2 Client Server

Client server is the server that needs to be monitored. The server is the location where the services which a customer has bought from Canter Oy, are running. This server is a virtual test server from VMware with operating system Windows Server 2008 R2 Standard 64-bit (6.1, Build 7601) and processor Intel® Xeon® CPU E5-2699 v3 @ 2.30 GHz, ~2.0 GHz and 4096 MB RAM. The server has Adeona JBoss and Adeona Salestool Test installed, where JBoss is the application service running the Adeona and Salestool is one of its plugins used to make online electronic catalogues, eCatalogs.

The list of running services can be seen in Figure 7. The Adeoa JBoss, Adeona Salestool Test and DB2 – AAMS – DB2 are the key services which maintain the Adeona product.

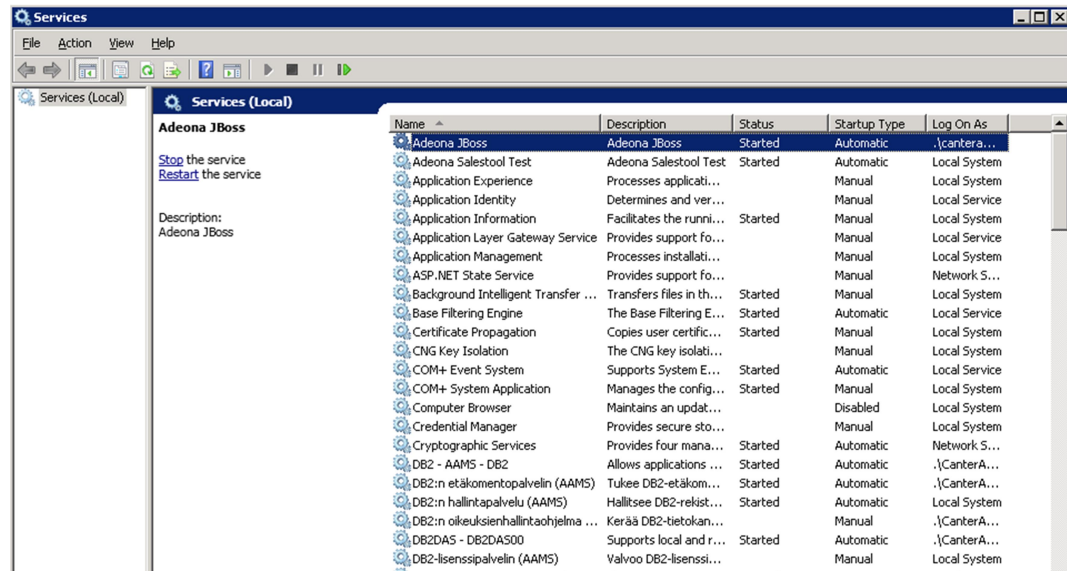


Figure 7. A list of some services on the server. Note the Adeona JBoss, Salestool Test and DB2 – AAMS – DB2

The ISP of this server is MPY and Canter Oy has full administrative rights on this server.

### 3 Powershell

Powershell has grown in the last few years and is still rapidly becoming more popular. The development of Powershell has greatly increased during these few years and the ability to include Linux bash inside it is the reason why it excels the others [2]. In short, Powershell is a scripting language built on .NET framework. It is object-oriented language which utilizes cmdlets to perform various tasks. These cmdlets are small programs that can be called directly from the command line or from a Powershell script file [3]. Every Windows operating system nowadays has a pre-installed Powershell.

Figure 8 shows the default layout of a Powershell command line and a basic command to check which chrome processes are currently running.

```

Windows PowerShell
PS D:\Powershell> Get-Process | Where-Object {$_.ProcessName -like "chrome"}

```

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	SI	ProcessName
257	24	45360	53116	270	2,31	988	1	chrome
246	25	51716	56644	249	2,53	1828	1	chrome
346	37	117884	122412	489	896,75	2540	1	chrome
241	26	55340	56912	244	26,81	3252	1	chrome
254	22	38836	26956	228	3,20	4364	1	chrome
3362	88	300268	150664	856	30,94	4640	1	chrome
240	22	38988	44800	228	0,73	4808	1	chrome
348	35	106148	82468	502	95,67	4908	1	chrome
690	46	182336	61044	488	9 804,31	5244	1	chrome
288	28	67652	41220	364	180,02	5404	1	chrome
290	27	53384	33060	364	35,22	6168	1	chrome
292	25	39420	24884	296	2,41	6544	1	chrome
347	30	53220	80908	361	43,09	9164	1	chrome
290	24	49892	20768	245	370,58	9352	1	chrome
257	25	47132	60736	253	2,70	9380	1	chrome
254	24	46584	44872	250	403,98	9780	1	chrome
294	29	77688	39180	387	135,08	9876	1	chrome
289	28	63560	50220	362	9,97	10348	1	chrome
285	27	61072	34544	349	156,95	11756	1	chrome
255	24	53252	35572	274	4,19	13672	1	chrome
294	27	51936	61008	344	2,08	15264	1	chrome

```

PS D:\Powershell>

```

Figure 8. The basic view of a Powershell command line and cmdlet

In Figure 8 the command, “Get-Process” is the cmdlet. If the command would have been run without the “Where-Object” statement, it would have displayed all of the running processes on the server. After the cmdlet is pipelined to the “Where-Object”, the desired filter can be set to Powershell to modify the results. In this example, the name of the process should be like “chrome”.

As stated above, Powershell has become more popular. A good server administrator should be able to write and read Powershell fluently since it will be the main tool for Windows in the future. Windows’ servers are becoming more GUI-less and this means that the role of the Powershell becomes more and more important.

For the sake of the present study it is important to explain the basics of the Powershell since the scripts used in this project needs to be analyzed. While technically the written scripts do not differ in any way in the example provided above, there will be more filtering, formatting and more complex commands. In the next chapter the scripts used in this project are briefly explained.

### 3.1 Scripts

This section describes the created scripts for monitoring purposes, their functionality and why they were designed the way they are. The scripts can be found in Appendix 1.

The scripts were designed in a way they would benefit both Canter and the customer. There are several basic monitoring functions, such as the disk size and the amount of CPU load on the server but also some unique functions specifically designed for the customer. In example, there have been cases where the customer has reported multiple performance issues with Adeona. Further investigations indicated that this was due to the fact that there were too many database connections opened to Adeona. Naturally, this increases the load of the database and leads to performance issues even though the application itself would work fine. It was decided that the connections to a specific port should also be monitored.

The customer also has several data integrations which can be monitored as well as the Adeona service which is running on the JBoss application server.

There have been cases where a malicious user has gained access to the server through a JBoss vulnerability. The attacker was able to set up a few bitcoin miners on the server. These miners were masked as regular windows processes which would turn on and off multiple times a day. It was discovered that the detection of the miners would become easier if the frequent changes in the state of the processes could be monitored.

#### 3.1.1 Properties

Like any other software, this monitoring system has its own configuration parameters. There is an option to turn some of the modules on or off, the “\$date” variable is used as a unique parameter to name the outgoing csv-files and the “\$csvdir” defines the path where these files are stored. Notice that the folder is pointing to the Dropbox folder.

Figure 9 shows the basic configurations of the monitoring system. User can choose which modules can be used when running the scripts.

```
$customer = "etra"  
$date = Get-Date -format "d_M_yyyy-HH_mm"  
$csvdir = "C:\Users\canteradmin\Dropbox\  
  
$prop_disks = "ON"  
$prop_connections = "ON"  
$prop_processes = "ON"  
$prop_systemtime = "ON"  
$prop_backups = "ON"  
$prop_integrations = "ON"  
$prop_jboss = "ON"  
$prop_changed = "ON"
```

Figure 9. Configurations in the script

Each of the available modules in Figure 9 will be explained during the next chapters.

### 3.1.2 Disk Space

Probably the most basic and common script is the disk space check. The basic structure of all the scripts is pretty much the same. First, the name of the output file is defined. This is done by using the "\$date" parameter with customer name, class of the script, which in this case is the disk space check, "disk" and the folder path. This process is done in all the scripts.

Figure 10 shows the complete command for getting the disk space. The most important parts of this script are explained.

```

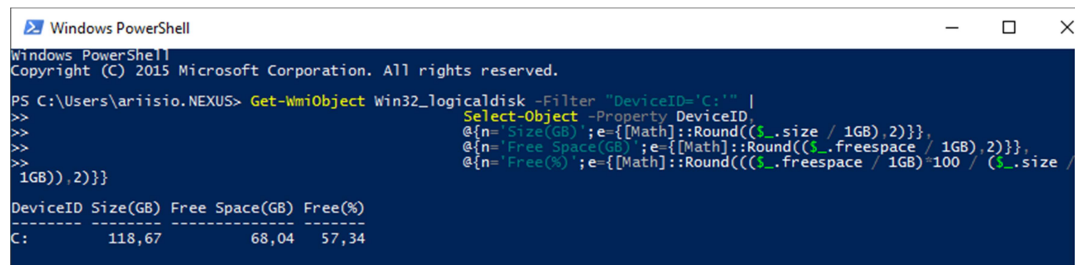
if ($prop_disks -eq "ON"){
try{
    $opt = "disks"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    Get-WmiObject Win32_logicaldisk -Filter "DeviceID='C:'" |
    Select-Object -
Property DeviceID,
@{n='Size(GB)';e={[Math]::Round((($_.size / 1GB),2)}}},
@{n='Free
Space(GB)';e={[Math]::Round((($_.freespace / 1GB),2)}}},
@{n='Free(%)';e={[Math]::Round(((($_.freespace / 1GB)*100 / ($_size /
1GB)),2)}}} |
Export-Csv $outfile -
NoTypeInfoation
}
catch{
Write-Error "Failed to get disk data from Etra Test
server"
}
}

```

Figure 10. The script used to fetch the disk space data

Note that the WmiObject is initialized and from which the win32\_logicaldisk class is used. It is notable that a new object is created and the items to this object are renamed and calculated, as line “@{n='Size(GB)';e={[Math]::Round(((\$\_.size / 1GB),2)}}” indicates. In this case the \$\_.size is a property of win32\_logicaldisk class and its value is formatted to gigabytes and rounded up by two decimals [4]. It is then renamed as “Size(GB)”.

Figure 11 shows the results for the command to fetch the disk space which can be run from any Powershell command prompt.



```

Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\ariisio.NEXUS> Get-WmiObject Win32_logicaldisk -Filter "DeviceID='C:'" |
Select-Object -Property DeviceID,
@{n='Size(GB)';e={[Math]::Round((($_.size / 1GB),2)}}},
@{n='Free Space(GB)';e={[Math]::Round((($_.freespace / 1GB),2)}}},
@{n='Free(%)';e={[Math]::Round(((($_.freespace / 1GB)*100 / ($_size /
1GB),2)}}}

DeviceID Size(GB) Free Space(GB) Free(%)
-----
C:        118,67      68,04      57,34

```

Figure 11. Results of the disk space check in Powershell window

This is how all the scripts work and it can be seen in the results above that the results are already in a form that they can directly be exported as csv.

### 3.1.3 Connections

The purpose of this script was to monitor the number of connections established to the Adeona database. To achieve this the best way is to use the `System.Net.NetworkInformation.IPGlobalProperties` object from .NET Framework [5].

The properties of the network are stored to an array which is then filtered using the port Adeona is using, 1098. Every connection to the port 1098 is stored to a new object as in previous script. This new object gets parameters such as local address and port, which are in fact the localhost address and the local port 1098, but also the remote address and port which means that the source of the connection can be seen.

Figure 12 shows the command for getting the number of connection to a certain port. Notable function is the `GetActiveTcpConnections` which stores every established connection in to an array.



```

if ($prop_connections -eq "ON"){
try {
    $opt = "connections"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $TCPPProperties =
[System.Net.NetworkInformation.IPGlobalProperties]::GetIPGlobalPropert
ies()
    $Connections = $TCPPProperties.GetActiveTcpConnections()
    $count = 0
    $localcon = 0
    $ArrList = [System.Collections.ArrayList]@()
    foreach($Connection in $Connections | where {$_.LocalEndPoint.Port
-eq 1098}) {
        $count++ | out-null
        if($Connection.LocalEndPoint.AddressFamily -eq "InterNetwork"
) { $IPType = "IPv4" } else { $IPType = "IPv6" }
        $OutputObj = New-Object -TypeName PSObject
        $OutputObj | Add-Member -MemberType NoteProperty
LocalAddress($Connection.LocalEndPoint.Address) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
LocalPort($Connection.LocalEndPoint.Port) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
ConnectedAddress($Connection.RemoteEndPoint.Address) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
RemotePort($Connection.RemoteEndPoint.Port) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
State($Connection.State) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
IPV4Or6($IPType) | out-null
        $ArrList.Add($OutputObj) | out-null
        if ($Connection.RemoteEndPoint.Address -eq
$Connection.LocalEndPoint.Address){
            $localcon++
        }
    }

    $result = $Arrlist | select LocalAddress, ConnectedAddress |
Export-Csv $outfile -NoTypeInfo
} catch {
    Write-Error "Failed to get active connections. $_"
}
}

```

Figure 12. Getting every connection to the server and looping them through and adding certain parameters if the family of the connection equals "InterNetwork"

It is important to gather the network information since the data can be utilized to see if there is any correlation between the network traffic and memory or cpu usage of Aedona.

### 3.1.4 Processes

One of the key factors in monitoring an environment is to analyze and measure the amount of processes and their performance. This is also a useful way to enhance reliability and security on the server since any suspicious or malicious processes can be seen if they are running on the server. This script contains two features. One to sum the memory and CPU consumption of all processes and the other to list each process. So, the output of this script is two separate csv-files.

Figure 8 shows the basic command to get information of the processes. To achieve better results, significant amount of formatting is needed. To measure the performance of all processes, working memory and cpu are used. However, the regular expression in Figure 8, "Get-Process | select CPU", only prints the number of seconds the specific process has used the processor, not the processor usage in percentages. The number of the regular CPU in the output might be misleading since a high CPU value does not necessarily mean high CPU usage.

Figure 13 shows how the CPU count is formatted to a more readable format. The total runtime of the process is related to the CPU load. Getting the results to satisfying format is easy and requires no special functions.

```
if($prop_processes -eq "ON"){
try{
    $opt = "processes"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $CPUPercent = @{n="CPUPercent";e={$TotalSec = (New-TimeSpan -
Start $_.StartTime).TotalSeconds
[Math]::Round(($_.CPU * 100 / $TotalSec), 2)}}

    $proc = Get-Process | Select-Object Name,@{n="Vir-
tualmemory(GB)";e={[Math]::Round((($_.VM / 1GB), 2)}),@{n="Working-
memory(GB)";e={ [Math]::Round((($_.WS / 1GB), 2)}),$CPUPercent, De-
scription | Sort CPUPercent -Descending
    $proc | Export-Csv $outfile -NoTypeInfoation
    $opt = "sumofprocesses"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
```

Figure 13. Using Powershell's math function

Figure 14 shows how the "measure-object" is used for example when summing each result from one column of the table.

```

    $sumofcpu = $proc | measure-object CPUPercent -Sum
    $sumofmem = $proc | measure-object "Workingmemory(GB)" -Sum
    $customtable = New-Object -TypeName PSobject
    $customtable | Add-Member -MemberType NoteProperty TotalCPU($sumofcpu.sum) | out-null
    $customtable | Add-Member -MemberType NoteProperty TotalMemory($sumofmem.sum) | out-null
    $customtable | Export-Csv $outfile -NoTypeInformation
  }
  catch{
    Write-Error "Failed to get processes on Etra Test
server"
  }
}

```

Figure 14. Using the Measure-Object cmdlet

Figure 13 only describes the monitoring of a single process but the information of the total CPU and memory consumption should be available. This can be done by using the `measure-object` cmdlet, which calculates specified numeric properties of an object. This can be seen in Figure 14. The results are stored to a different CSV-file, since the information is imported to a different table in the database.

### 3.1.5 System Time

The time the server has been powered on can sometimes be useful information as well as the previous time of a reboot. Powershell can record these easily by using the `Get-WmiObject` cmdlet and its class called `Win32_operatingsystem`. From this class, only the last boot time is needed since the uptime of the server can be calculated by subtracting the last boot time from current date. Both the uptime and the last boot time are recorded to the same CSV-file.

Figure 15 shows the complete script for getting the information. Powershell has direct function to check how long the machine has been on.

```

if($prop_systemtime -eq "ON"){
try{
    $opt = "systemtime"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $system = Get-WmiObject Win32_operatingsystem
    $up = (Get-Date) - ($system.ConvertToDateTime($system.last-
bootuptime))
    $boottime = $system.ConvertToDateTime($system.lastbootuptime)
    $customsystem = New-Object -TypeName PSobject
    $customsystem | Add-Member -MemberType NoteProperty Up-
time("$($up.Days) days, $($up.Hours) hours, $($up.Minutes) minutes")
    $customsystem | Add-Member -MemberType NoteProperty
BootTime($boottime)
    $customsystem | Export-Csv $outfile -NoTypeInformation
    }
    catch{
        Write-Error "Failed to get systemtime on Etra Test
server"
    }
}
}

```

Figure 15. Script to calculate the time system has been on

Knowing how long the server has been on is crucial since servers should be on constantly. Only a few times a year, boot and update is required during a scheduled maintenance break.

### 3.1.6 Backups

Like every other server with a database, etradeona01 has backups. These backups are essential since they contain all the product data the customer has in their database. If an issue emerges with the database and for example the whole database is corrupted it is important that the backups are present on the server so that they can be rolled back. Therefore, it is necessary that these backups are being monitored.

Figure 16 displays the script to check if the backups exist. In Powershell, the Get-Date works in a way that if a date in example 5 days ago needs to be set, then -5 days are added to the cmdlet.

```

if($prop_backups -eq "ON"){
try{

    $opt = "backups"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $limit = (Get-Date).AddDays(-1)
    $results = Get-Childitem -Path C:\TESTDB2BACKUP -Recurse -
Force | where-object {$_.CreationTime -gt $limit }
    if ($results){
        $backup = $True
    }else{
        $backup = $False
    }

    $amount = (Get-Childitem -Path C:\TESTDB2BACKUP -Recurse -
Force | Measure-Object).Count

    $customback = New-Object -TypeName PSObject
    $customback | Add-Member -MemberType NoteProperty
backup_name($results)
    $customback | Add-Member -MemberType NoteProperty
backup_found($backup)
    $customback | Add-Member -MemberType NoteProperty total_num-
ber_of_backups($amount)
    $customback | Export-Csv $outfile -NoTypeInformation
    }
    catch{
        Write-Error "Failed to get backups on Etra Test
server"
    }
}
}

```

Figure 16. Filtering the script to only handle backups from day before

The database on the server stores backups to a specified folder which in this case is "C:\TESTDB2BACKUP". The limit shown in Figure 16 is set as one day before current date. The script returns a boolean value whether backup is found or not from the previous day. It also calculates the number of total backups found from this folder. Since this folder is only for backups, there is no need to do any specific file naming or folder filtering.

Figure 17 indicates the output if the script in Figure 16 is run directly from the Powershell.



```

Windows PowerShell
PS C:\PowershellScripts> .\get_backups.ps1
backup_name                backup_found                total_number_of_backups
-----
RAMST_0.DB2.NODE0000.CATN0000.201606... True                          33
PS C:\PowershellScripts> _

```

Figure 17. The output of the backups monitoring script

As usual, the information is stored to a CSV-file, which is then stored to a database.

### 3.1.7 Integrations

As a product information company, Canter deals with remarkable amounts of product data. One of the challenges is to transfer this data from a customers' environment to the Adeona database. Most of the data comes directly from a customers' ERP to a specified server. These servers are mostly running on a Windows operating system and the data is integrated to a MSSQL database.

Another phase is to get the product data to be displayed on a website. This is completely different integration which depends on who is managing the customer's web shop. In this case, the focus is on the ERP-Adeona integration.

There are many variables regarding the data transformation. The product data itself is usually transferred with xml-files or using REST API queries. The product images are transferred separately, usually using rsync or ftp-transformation. Whether the product data is transferred via REST queries or xml-files the integration always logs the process to a log file specified by the log4j.xml together with Kettle.

Figure 18 shows the configuration currently in use in log4j software.

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
3
4  <log4j:configuration>
5
6  <appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
7    <layout class="org.apache.log4j.PatternLayout">
8      <!--<param name="ConversionPattern" value="%r [%t] %-5p %c %x - %m\n" /-->
9      <!--<param name="ConversionPattern" value="%d{dd MM yyyy HH:mm:ss,SSS} %L %-5p %20c - %m\n" /-->
10     <param name="ConversionPattern" value="%d{HH:mm:ss,SSS} %L %-5p [%c(1)] - %m\n"/>
11   </layout>
12 </appender>
13
14
15 <!-- A size based file rolling appender -->
16 <appender name="FILE" class="org.apache.log4j.DailyRollingFileAppender">
17   <param name="File" value="logs/client.log"/>
18   <param name="Append" value="true"/>
19   <param name="DatePattern" value="yyyy-MM-dd"/>
20   <layout class="org.apache.log4j.PatternLayout">
21     <param name="ConversionPattern" value="%d %-5p [%c] %m\n"/>
22   </layout>
23 </appender>
24
25
26 <root>
27   <priority value="info" />
28   <appender-ref ref="ConsoleAppender"/>
29   <appender-ref ref="FILE"/>
30 </root>
31
32 </log4j:configuration>

```

Figure 18. Example log4j configuration

The product data of this customer is transferred by copying the ERP database directly to the Adeona server, then using a Kettle plugin to execute three different phases:

1. erp2xml. Generates an xml file from the products that match the condition whether a product should be moved or not. Naturally, this saves time by not moving all of the products
2. xml2aams. Generates the product data hierarchy from the ERP database. This is so that the products and their data find their position in the product tree.
3. data2aams. Transfers the data using temporary database tables and constructs the data based on the previous steps. These temporary tables are then switched to production to update the data to web.

Each of these steps generate their own log file. To be ascertain that the data has successfully been transferred, each of these files needs to be analyzed for errors.



A complete list of all the error messages which are parsed from the log files can be seen in Appendix 1. If an error occurs during any of the steps below, Kettle and log4j will record this incident to the log file. The proper log file is found by filtering the contents of the log folder by selecting only the latest file. This file is then parsed using the parameters in the array list.

Figure 19 shows how the script searches for the log file and analyzes it from the given keyword in the \$errors array.

```

if($prop_integrations -eq "ON"){
try{
    $opt = "integrations"
    $errors = @("Finished with errors", "error", "Error")
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $limit = (Get-Date).AddDays(-1)
    $integrations = [System.Collections.ArrayList]@()
    $logfile = "C:\integrations\etra_erp2aams\data2aams\logs"
    $check = Get-Childitem -Path $logfile | Where-Object {$_.Last-
writetime -gt $limit -and $_.extension -eq ".log"}
    if ($check){
        $status = $check | foreach-object {if (select-string -
pattern $errors "$logfile\$_"){
            $error = $True
        }else{
            $error = $False
        }
    }
}
}
}

```

Figure 19. Setting the limit for log files to be searched.

If any of the parameters is found, the boolean value is stored to \$error variable. Depending on the value of this variable certain actions are made. \$customint object is initialized and members are added to it. The values of these members vary if the boolean value of \$error variable is true or false. This same process is repeated to the other two parts of the integration.

### 3.1.8 Jboss

One of the key elements of this monitoring system, is to monitor the Adeona service itself. The service runs on an application server JBoss as a windows service.

Figure 20 shows the Powershell's own built-in module, Get-Service, for checking services and their statuses on a machine.



```

Windows PowerShell
PS C:\> get-service | Where-Object {$_.name -like "w*"}
Status Name DisplayName
-----
Stopped W32Time Windows Time
Stopped WalletService WalletService
Stopped wbcengine Block Level Backup Engine Service
Stopped WbioSrv Windows Biometric Service
Running Wcmsvc Windows Connection Manager
Stopped wcnscvc Windows Connect Now - Config Registrar
Running WdiServiceHost Diagnostic Service Host
Running WdiSystemHost Diagnostic System Host
Running WdNisSvc Windows Defender Network Inspection...
Stopped WebClient WebClient
Stopped Wecsvc Windows Event Collector
Stopped WEPHOSTSVC Windows Encryption Provider Host Se...
Stopped wercplsupport Problem Reports and Solutions Contr...
Stopped WerSvc Windows Error Reporting Service
Stopped WiaRpc Still Image Acquisition Events
Running WinDefend Windows Defender Service
Running WinHttpAutoProx... WinHTTP Web Proxy Auto-Discovery Se...
Running Winmgmt Windows Management Instrumentation
Stopped WinRM Windows Remote Management (WS-Manag...
Stopped wisvc Windows Insider Service
Running WlanSvc WLAN AutoConfig
Stopped wldsvc Microsoft Account Sign-in Assistant
Running wltrysvc DW WLAN Tray Service
Stopped wmiApSrv WMI Performance Adapter
Stopped WMPNetworkSvc Windows Media Player Network Sharin...
Stopped workFolderssvc Work Folders
Stopped WPDBusEnum Portable Device Enumerator Service
Running WpnService Windows Push Notifications System S...
Stopped WpnUserService... Windows Push Notifications User Ser...
Running wscsvc Security Center
Running WSearch Windows Search
Stopped wuauclt Windows Update
Running wudfsvc Windows Driver Foundation - User-mo...
Stopped WwanSvc WWAN AutoConfig
PS C:\>

```

Figure 20. Example Powershell command of checking services

The name of the service on the server is “Adeona JBoss”. On page 5 in Appendix 1, only name and status of this service is checked with the Get-Service method.

### 3.1.9 Changed Processes

The script in Appendix 1 page 6 was designed to detect processes flapping too often. At the time of writing the study, some of the servers were suffering from a vulnerability in the JBoss application server. This allowed BitCoin miners to infiltrate the server and use the server’s CPU almost completely. These Bitcoin miners were easy to detect since they were over flapping which meant that they were constantly turning off and on. They were also consuming high amount of CPU and were usually masked as a regular windows process.

Since most of the processes running on the server were stable, the script could be made. If there would have been multiple processes running on the server which would be turning on and off, the data would have been hard to read. But in this case, this script became a useful tool at detecting ambiguous processes.

The main functionality of the script is based on comparing the running processes between timeframes. If the Powershell monitoring system is set to run in 15 minute intervals, then this timeframe is 15 minutes. Before this script is being run, there is already a csv-file on the server, called "initial\_process\_list.csv". The script then fetches the current running processes to a table and compares this table to the existing csv-file. Powershell's Compare-Object returns a string if there are differences between these tables and this value can then be used to indicate if there are new processes which were not running 15 minutes ago.

The table to which the current running processes were saved is then stored over the "initial\_process\_list.csv" file which is again used when the script is being run next time.

Figure 21 shows the functionality of the script comparing overlapping processes.

```

Get-Process | Select-Object Name | Export-Csv C:\Power-
Mon\files\process_comparator.csv
$a = Get-Content C:\PowerMon\files\initial_process_list.csv
$b = Get-Content C:\PowerMon\files\process_comparator.csv
$arr = [System.Collections.ArrayList]@()
$rows = Compare-Object $a $b

if($rows){
    foreach ($row in $rows){
        if($row.SideIndicator -eq "=>"){
            $state = "NEW"
        }
        if($row.SideIndicator -eq "<="){
            $state = "OLD"
        }

        $changed = New-Object -TypeName PSObject
        $changed | Add-Member -MemberType NoteProperty
Name($row.InputObject)
        $changed | Add-Member -MemberType NoteProperty
State($state)

        $arr.Add($changed) | Out-Null
    }
    $arr | Export-Csv $outfile -NoTypeInformation
Remove-Item C:\PowerMon\files\initial_pro-
cess_list.csv -force
    Rename-Item C:\PowerMon\files\process_compara-
tor.csv -NewName initial_process_list.csv
    }
}

```

Figure 21. Main functionality of the "changed processes" script

In Figure 21, the SideIndicator option is used to detect whether the value existed in the previous list of processes or not. If the SideIndicator equals “=>” it means that the process was not found in the list and therefore the value “NEW” is placed in the array.

## 4 Configuration of Environments

This chapter goes through the configuration process of the client and the host server. Multiple configurations are needed so that the system works fluently. First, the Powershell needs to be configured so that the scripts can be run automatically and the host server needs to be set up so that the data is transferred.

### 4.1 Client Server

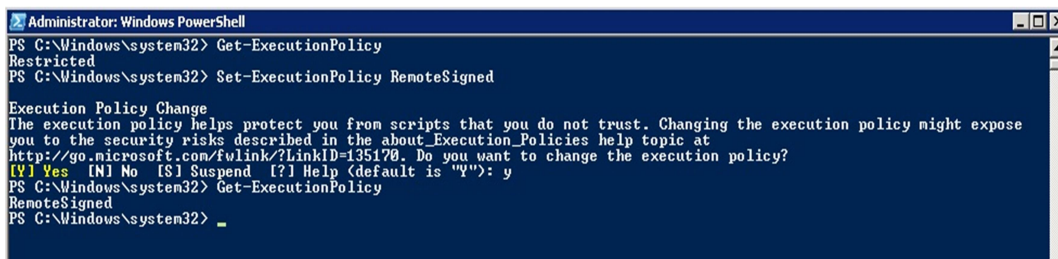
The only installation required on the Windows Server is the Dropbox client. The scripts run through the Windows Task Scheduler and the results are saved as CSV-files in the home folder of the Dropbox client.

In the present study, the Dropbox client was downloaded from <https://www.dropbox.com/install> and the Dropbox folder was installed to the path C:\Users\canteradmin\Dropbox. The Dropbox was installed as an administrator which prevents other users accessing the folder which is synchronized to the cloud.

Running Powershell scripts through the Task Scheduler is easy but there are a few things which need to be done beforehand. First, Powershell by default works in a way that it prevents scripts running automatically for security reasons. It is possible to adjust the settings and allow the scripts to be run locally or even remotely. Since Powershell runs locally on the client's server only the first option is needed [6].

By opening Powershell as an administrator and executing the following command: Set-ExecutionPolicy remotesigned the scripts can be allowed to be run through Task Scheduler.

Figure 22 shows how the execution policy can be changed from the Powershell.



```
Administrator: Windows PowerShell
PS C:\Windows\system32> Get-ExecutionPolicy
Restricted
PS C:\Windows\system32> Set-ExecutionPolicy RemoteSigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about_Execution_Policies help topic at
http://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y
PS C:\Windows\system32> Get-ExecutionPolicy
RemoteSigned
PS C:\Windows\system32> _
```

Figure 22. Checking and setting the execution policy in Powershell

Now a task from the task scheduler can be created. When running Powershell scripts through the Task scheduler it is not enough to just define the action by pointing to the path where the script is located as it would be when running regular Windows batch scripts. First, the path where Powershell is located is needed and then an argument must be added to point out the Powershell script location.

Figure 23 indicates how the task scheduler is configured using the “Add arguments” option.

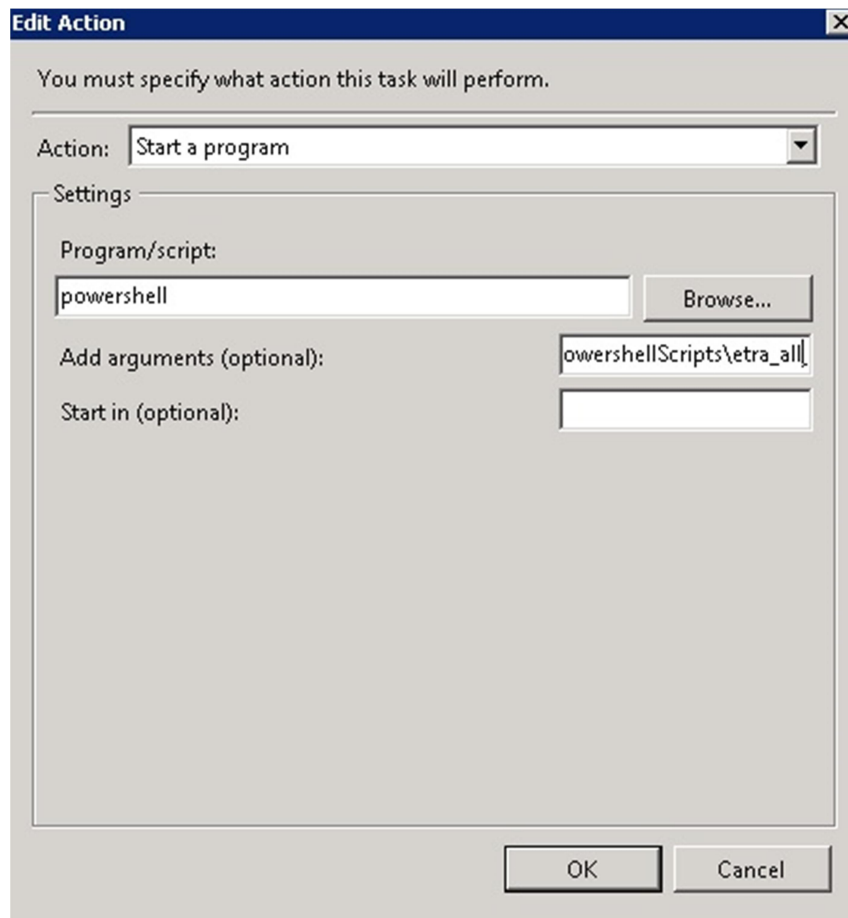


Figure 23. Setting up the location of the Powershell script

Naturally, the script can be scheduled at any time. For the present study, the tasks were scheduled to run at every 15 minutes. This timeframe is enough to provide real time information from the server.

No other settings or configurations are needed on the client's server. Once these installations are done the blank Ubuntu Server can be set up.

## 4.2 Ubuntu Server

The following chapters describe how the LAMP and its configurations, scripts for fetching and storing the collected data are set up. One of the most considerable challenges was to find the proper configurations for the packages in order for this system to work properly.

#### 4.2.1 Dropbox Client

The following are the instructions how to set up the Dropbox client on the Ubuntu Server [7]. This is a “GUI-less” installation since the server does not have graphical user interface. It should also be mentioned that after all the configurations were done and the necessary updates were installed the csv-files were unable to sync. Unnecessary hours were spent to debug this issue and it was found that there was an issue in the latest Dropbox version. This version needed to be downgraded and therefore these installation instructions are for the older version:

First, the required dependencies needs to be installed. These are:

- Graphical user interface library, libgtk2.0-0
- Session management library, libsm6

To install these, following commands are run from the command line:

- `sudo apt-get install libgtk2.0-0`
- `sudo apt-get install libsm6`

Then the Dropbox client:

- `cd ~ && wget -O - "https://dl.dropboxusercontent.com/u/17/dropbox-lnx.x86_64-2.10.51.tar.gz" | tar xzf -`

This installs the Dropbox client and after the installation is finished the created daemon needs to be connected to the Dropbox account:

- `~/dropbox-dist/dropboxd`

This command prompts the user to input a working link from the Dropbox account to the command line. After the link has been established, the installation of the Dropbox is finished and every time when something is moved to the Dropbox folder on the client's server the content is transferred to the Ubuntu Server.

Dropbox client works in a way that it synchronizes the content between two folders on two different servers. If it is assumed that the Dropbox folder on the client's server is called A, and B is the folder on our Ubuntu Server and if some text files are moved to A, in a short time they are synced to B. Now if these synced text files are moved away from B to a different folder, i.e B\_test, the files are removed from A as well.

This is a very good functionality since this can increase the confidentiality of the data by constantly running a command on the Ubuntu server which moves all the files on the folder to another location on the server. This way, if the Dropbox client is somehow hacked, the data for the attacker is not constantly available.

Cron can be used to run simple move command. The PHP-script in the screenshot is explained later. In this thesis, the timeframe to move the files from the Dropbox folder is 15 minutes but it can be configured to run this command every minute if necessary.

Figure 24 shows how the Cron is set up on the Ubuntu Server.

```

# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/15 * * * * /bin/mv ~/Dropbox/e* /home/psdownloads/ > /dev/null 2>&1
*/15 * * * * /usr/bin/php /home/phpscripts/get_files_dev.php > /dev/null 2>&1

```

Figure 24. A command to move the contents of the Dropbox folder to another location

The contents of the Dropbox folder are transferred to /home/psdownloads/ and they are now ready to be processed.

#### 4.2.2 Installing LAMP

As mentioned earlier, as the files have been transferred between the two, or more, servers they need to be processed. Since the files are in csv-format it is easy to import them to a mysql database using PHP-script. The database is needed because the gathered information needs to be displayed on a website running on Apache2 web-server.

The installation of LAMP is very straightforward and does not need further discussion. Installation instruction for LAMP can be found at: <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu>

However, there are several settings which are needed in order to work this system properly. First of all, it needs to be made sure that the bind address for mysql is set to localhost only for security reasons. The default path for mysql configuration file is `/etc/mysql/my.cnf`

In this file, the following line has to be located ensuring that the value is set to 0.0.0.0

- `bind-address = 0.0.0.0`

The `local_infile` variable to this configuration file needs to be defined. Since automated PHP-scripts are used to load the csv files to the database, it is important that this is configured. The `local_infile` variable defines whether the “LOAD\_DATA\_INFILE” command can be used in the PHP-scripts or not. To set this variable, the following line should be added under “Basic settings” to the `my.cnf` file:

- `local-infile=1`

There are also a few settings needed for PHP. The main tool for connecting to the mysql database in php is `mysqli`. This needs to be enabled and to do this, certain lines needs to be uncommented from php configuration file which by default is located in `/etc/php5/apache2/php.ini`. The following line under “mysqli” needs to be located and uncommented by removing the semicolon in front of the line.

- `mysqli.allow_local_infile = On`

After these settings have been enabled, the virtual host for Apache can be set. Only a few configurations are needed to make this system more secure and stable.

Default configurations for virtual hosts can be found from `/etc/apache2/sites-available/`. There is a default configuration file “000-default.conf” which can be used. The file is copied by running a command:



- `sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/powermon.fi.conf`

Figure 25 shows what configurations are needed after the copied file is opened.

```
<VirtualHost *:8080>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin aki.riisio@canter.fi
ServerName powermon.fi
ServerAlias www.powermon.fi
DocumentRoot /var/www/powermon.fi/public_html
ErrorLog ${APACHE_LOG_DIR}/powermon_error.log
CustomLog ${APACHE_LOG_DIR}/powermon_access.log combined

<Directory /var/www/powermon.fi/public_html>
    Order Allow,Deny
    Allow from all
</Directory>

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Figure 25. Virtualhost configuration.

The website itself will be added later to the directory in this configuration. The next section covers the part of designing and creating the database.

#### 4.2.3 Database

In the previous section the installation of the LAMP and the necessary configurations for the mysql were described. In this section, the architecture of the database is covered.

As stated above, the LAMP installation includes the MySQL database. Even though the database is relational, for the purpose of the present study relational database would not be necessary. However, the MySQL database performs very well on a Linux environment and has very few issues regarding importing and exporting data to the application layer.

The database was designed in a way that every customer would have their own database. And as can be seen from the scripts, the results of each monitored target are saved to their separate csv files. Each of these csv files is then imported to the customer's database to their own representative tables. For the sake of the present study, only one database and customer is covered.

Figure 26 shows all the related tables for one customer. Name of the table should always start with the name of the customer.



```

root@canteradmin:~
root@canteradmin:~# mysql -u root -psql5my
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 38
Server version: 5.5.43-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| etra_power |
| mysql |
| performance_schema |
| testdb |
+-----+
5 rows in set (0.05 sec)

mysql> use etra_power;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables in etra_power |
+-----+
| etra_adeonajboss |
| etra_backups |
| etra_changedprocesses |
| etra_connections |
| etra_disks |
| etra_integrations |
| etra_processes |
| etra_sumofprocesses |
| etra_systemtime |
+-----+
9 rows in set (0.00 sec)

mysql>

```

Figure 26. Customer's database and associated tables

Since there might be more than one customer using the system the tables should be created automatically. This is done by a PHP-script described in the next chapter.

#### 4.2.4 Generating Tables

The script in Appendix 2 is designed to quickly install the customer to the system. The script is specifically for customer Etra, but it can easily be expanded by replacing the static customer with a parameter. This way, the script could be directly run from the command line while giving only the customer name as a parameter.

Figure 27 shows how the MySQL connector is initialized. This is done by using the “mysqli” extension available in MySQL [8]. This has a direct support to PHP.

```
$server = "localhost";
$username = "root";
$password = "*****";
$db = "etra_power";

$conn = new mysqli($server, $username, $password, $db);
if ($conn->connect_error){
    die("Connection failed: " . $conn->connect_error);
}
```

Figure 27. Initializing the MySQL connector

When the connector has been assigned to a variable, it can be called throughout the whole script to quickly access the database.

Then each of the table creation scripts are stored as strings to specified variables which are called when the connection is made to the database. These strings contain the datatypes for certain fields. For instance, if the amount of memory a process is consuming is stored to a table it is wise to use DOUBLE(10,2). This makes sure the decimals are stored.

Figure 28 indicates what kind of datatypes are needed.

```

$sql_disks = "CREATE TABLE IF NOT EXISTS etra_disks (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    device_id VARCHAR(5),
    size DOUBLE(10,2),
    free_space DOUBLE(10,2),
    free_percent DOUBLE(10,2),
    timestamp DATETIME
)";

```

Figure 28. SQL script to create the table for storing disk information

The same principle applies to all the other tables. These SQL-scripts are then executed using the “mysql” connector in the following way described in Figure 29.

```

    if ($conn->query($sql_disks) === TRUE){
        echo "Table etra_disks created succesfully\n";
    }else{
        echo "Error creating table etra_disks:\n " . $conn-
>error;
    }

```

Figure 29. Executing the SQL script

If the connection is successful, the table is created and can be used to store information.

#### 4.2.5 Automating Data Flow

As it was previously stated, the CSV files collected from the customer’s server flow to the Dropbox folder on Ubuntu server. The files are then moved to a better location from which they are stored to the specified database. This is done by a PHP-script which is scheduled to run every 15 minutes. The syntax in CRON can be seen in Figure 24.

First, every file needs to be assigned to a variable for proper handling. Using the glob() function from PHP library the files can easily be checked as to whether they exist or not. Definition of the location of the files can be seen in Figure 30.

```

error_reporting( error_reporting() & ~E_NOTICE );

$file_disks = '/home/psdownloads/etra_disks_*';
$file_processes = '/home/psdownloads/etra_processes_*';
$file_connections = '/home/psdownloads/etra_connections_*';
$file_sumproc = '/home/psdownloads/etra_sumofprocesses_*';
$file_systemtime = '/home/psdownloads/etra_systemtime_*';
$file_backups = '/home/psdownloads/etra_backups_*';
$file_integrations = '/home/psdownloads/etra_integrations_*';
$file_jboss = '/home/psdownloads/etra_adeonajboss_*';
$file_changed = '/home/psdownloads/etra_changed_processes_*';

```

Figure 30. Storing the files to a variable.

Figure 31 shows how the glob() function is used.

```

foreach (glob($file_disks) as $etra_diskfile){
    echo "File $etra_diskfile found!\n";
}

```

Figure 31. Checking if the file exists.

The script in Figure 31 tries to find if a file exists in location defined in a variable `$file_disks` and if it is found, it will be stored to a variable called "`$etra_diskfile`". When the process has been done to all of the required files, the `mysqli-connector` needs to be set up.

Note that a different connector is used. This is because when importing csv-files to a database table, certain options need to be set before importing. This could have not been achieved using the regular `mysqli` connector above. Instead, `mysqli_options` and `mysqli_real_connect` are required [9].

Figure 32 shows the required configurations for the `mysqli_options` and `mysqli_real_connect`.

```

$user = 'root';
$pass = '*****';
$db = 'etra_power';
$host = 'localhost';

$mysqli = mysqli_init();

mysqli_options($mysqli, MYSQLI_OPT_LOCAL_INFILE, true);
#Initialize the database connection
mysqli_real_connect($mysqli,$host, $user, $pass,$db)
    or die ('<P>ERROR connecting to $db</P>');|

```

Figure 32. Setting the options for mysqli connection

The option “MYSQLI\_OPT\_LOCAL\_INFILE, true” is related to the setting Covered in chapter 4.2.2. Setting the option enables reading local files.

After the required options are set the query can be initialized. The same method is used as previously: the SQL-query is stored as a string to a variable. Notice the local infile load.

Figure 33 shows how the “LOCAL INFILE” is used in the SQL script.

```

$query_disks = "LOAD DATA LOCAL INFILE '$etra_diskfile'
    INTO TABLE etra_disks
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (device_id,size,free_space,free_percent)
    SET timestamp=NOW() ";

```

Figure 33. Initialization of the SQL-script

Since the file to be imported is CSV, the script needs to know the delimiters and line terminations. These are set in the script above as well as the header line.

The script can then be executed using the regular mysqli-connector. If the CSV-file includes all the required columns the import should be successful.

Figure 34 shows how the SQL query in Figure 33 is executed.

```

if ($mysqli->query($query_disks) === TRUE){
    echo "Data imported succesfully\n";
}else{
    echo "Error importing data: \n" . $mysqli->er-
ror;
}

```

Figure 34. Importing the CSV-file to the database.

Figure 35 displays the results which can be verified directly from the MySQL-database. At the time the following script as seen in Figure 34 was run, the system had been running for 5 days.

```

mysql> select * from etra_disks where id > 500;
+-----+-----+-----+-----+-----+-----+
| id | device_id | size | free_space | free_percent | timestamp |
+-----+-----+-----+-----+-----+-----+
| 501 | C: | 199.00 | 13.00 | 6.00 | 2016-02-22 08:45:01 |
| 502 | C: | 199.00 | 13.00 | 6.00 | 2016-02-22 09:00:01 |
| 503 | C: | 199.00 | 13.00 | 6.00 | 2016-02-22 09:15:01 |
| 504 | C: | 199.00 | 13.00 | 6.00 | 2016-02-22 09:30:01 |
| 505 | C: | 199.00 | 13.00 | 6.00 | 2016-02-22 09:45:01 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> █

```

Figure 35. The results from the etra\_disks table.

It was thought that it would be convenient to archive the stored CSV-files. In the future, this could be an optional parameter but in this case the archiving logic was included in the script. Figure 36 shows how the logic works after the data has been imported to the database.

```

$mysqli->close();

$files = scandir("/home/psdownloads");
$source = "/home/psdownloads/";
$destination = "/home/psarchive/";
foreach ($files as $file) {
    if (in_array($file, array(".", ".."))) continue;
    if (copy($source.$file, $destination.$file)) {
        $delete[] = $source.$file;
    }
}
foreach ($delete as $file) {
    unlink($file);
}

```

Figure 36. Archiving the CSV-files.

After the database connection is closed, the folder where the files were originally stored is scanned using the scandir-function. The results can then be looped through and if a file is found from the source path it will be moved to another location. The file is then deleted from the source folder so that it will not intervene with the next files to be imported.

## 5 Website

The intention was to create a simple website to display the data collected from the server. While the basic principle was achieved, the amount of work grew too high and while the website was up and running it did not turn out the way it was intended.

In the following chapters the basic functionality is covered and the layout is presented. The website can be divided into three parts:

- Home page
- Data table page which shows the data in tables
- Graphic page which displays a chart generated by JPGraph

The source code for the pages can be found in Appendixes 4, 5 and 6. For the sake of the present study, only the graph and datatables page are covered.

### 5.1 Data Tables Page

This data tables page has sections where the user can select the appropriate timeframe in which the data is displayed. The number of sections or tables on this page equals the number of tables in the database. By default, the timeframe is set as "Now". And as can be seen in Figure 37, if there are no data in the given timeframe the page displays a small alert under the table. If the timeframe is set as "Now", it will try to find all data during the last 15 minutes.



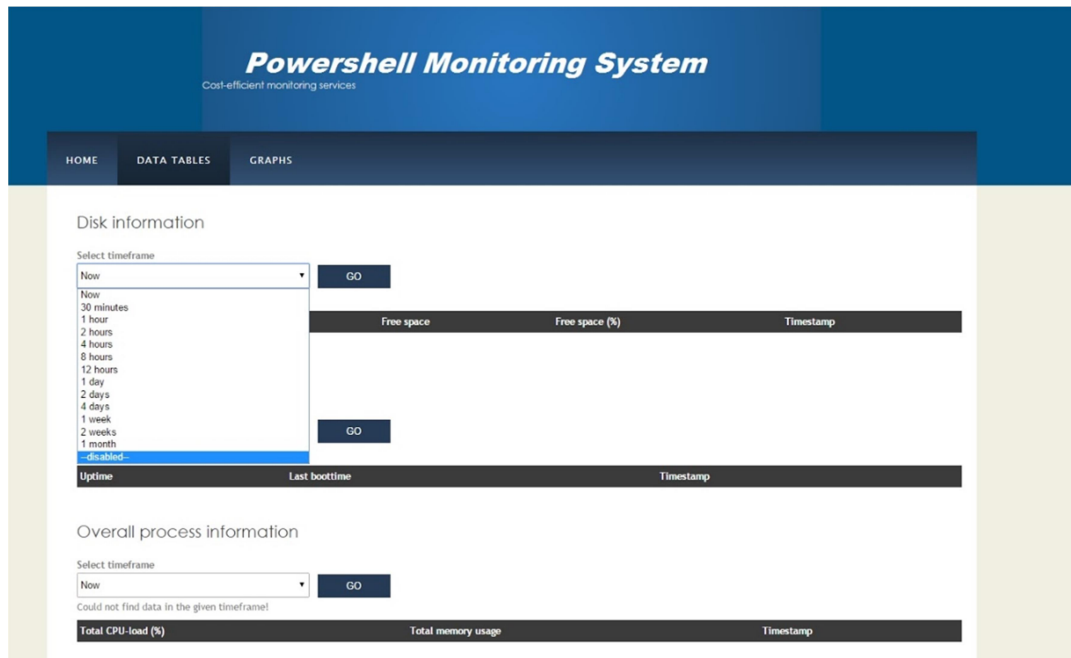


Figure 37. Layout and the dropdown menu on the page

The list of time options is created with a PHP-script embedded on the page with a function called `createDropdown`. This creates the dropdown menu together with the `create_form` function. If the user selects a value from the generated list, it is passed to a variable which is used in a SQL-script to load the data. The script for this process can be seen in Figure 38.

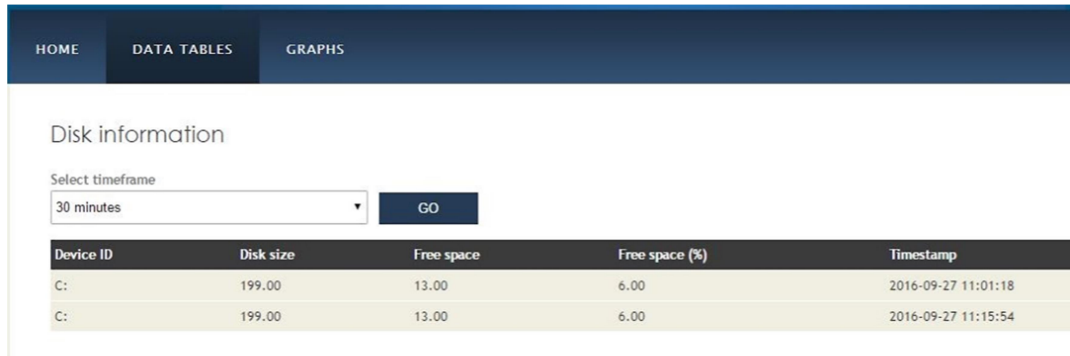
```
<?php createDropdown($x1); ?>
<p style="padding-right: 15px"><span>&nbsp;</span><input class="submit" type="submit" name="name" value="GO" /></p>

<?php

if($x1 == 1){ $time1 = 'NOW() - INTERVAL 15
MINUTE'; };
if($x1 == 2){ $time1 = 'NOW() - INTERVAL 30
MINUTE'; };
if($x1 == 3){ $time1 = "NOW() - INTERVAL 1
HOUR"; };
if($x1 == 4){ $time1 = "NOW() - INTERVAL 2
HOUR"; };
```

Figure 38. Selecting the timeframe to be used in the SQL-script

If the user selects a timeframe which has data, the data will be displayed on the page. If no data is found in the given timeframe a notification is displayed. Figure 39 shows how the data is displayed.



Device ID	Disk size	Free space	Free space (%)	Timestamp
C:	199.00	13.00	6.00	2016-09-27 11:01:18
C:	199.00	13.00	6.00	2016-09-27 11:15:54

Figure 39. Results when selecting data

The table illustrated in Figure 39 is generated inside the web page. The following method (see Figure 40) is used in each of the monitored service.

```

        if($time1 == 'DISABLED'){
            echo "<b>Option disabled</b>";
        }else{
            echo "<table style='width:100%; border-spacing:0;'>
                <tr>
                    <th>Device ID</th>
                    <th>Disk size</th>
                    <th>Free space</th>
                    <th>Free space (%)</th>
                    <th>Timestamp</th>
                </tr>";

            if ($result->num_rows > 0){
                // output data of each row
                while($row = $result->fetch_assoc()){
                    echo "<tr>";
                    echo "<td>" . $row['device_id'] .
" </td>";
                    echo "<td>" . $row['size'] . " </td>";
                    echo "<td>" . $row['free_space'] .
" </td>";
                    echo "<td>" . $row['free_percent'] .
" </td>";
                    echo "<td>" . $row['timestamp'] .
" </td>";
                }
            }else{
                echo "Could not find data in the given
timeframe!";
            }

            echo "</table>";
        }

        $conn->close();
    }
}

```

Figure 40. Generating the table to display the results.

In some cases, it would be convenient to see a graphical representation of the results. For instance, comparing the CPU load and memory consumption could be compared properly if the information was available in line graphs. This is done by using the JpGraph which is a library for PHP. This is described in the next chapter

## 5.2 JpGraph

JpGraph is a useful tool for creating simple but efficient charts embedded to a PHP page. Example plot in JpGraph can be seen in Figure 41.

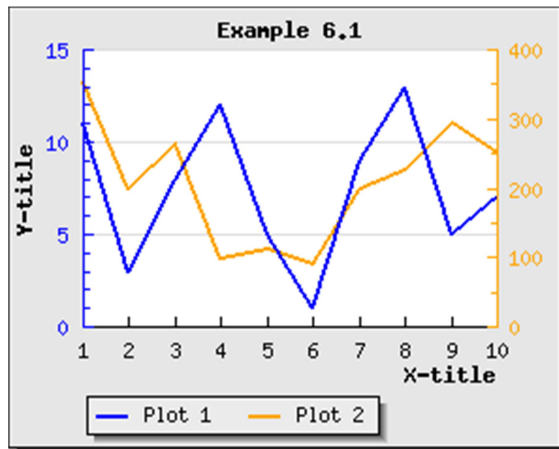


Figure 41. Example of a JGraph plot

On the Graphs page (see Appendix 6), the plot has a separate PHP-script which is called when a user selects suitable time frame. The method is the same when importing data to a table in the datatables page. When the script is called it generates an image which can be embedded to the page. Figure 42 shows how the data for x- and y-axes are set.

```

$ssql_sumofprocesses = "SELECT total_cpu FROM etra_sumofprocesses WHERE timestamp >= $timel;";
$ssql_hour = "SELECT DATE_FORMAT(timestamp, '%H:%i:%s') FROM etra_sumofprocesses WHERE timestamp >= $timel;";
$result = $conn->query($ssql_sumofprocesses);
$test_data = array();
if ($result->num_rows > 0){
    // output data of each row
    while($row = $result->fetch_assoc()){
        $test_data[] = $row;
    }
}
else{
    echo "Could not find data in the given timeframe!";
}

$result = $conn->query($ssql_hour);
$test_data2 = array();
if ($result->num_rows > 0){
    // output data of each row
    while($row = $result->fetch_assoc()){
        $test_data2[] = $row;
    }
}
else{
    echo "Could not find data in the given timeframe!";
}

$conn->close();

$yaxis = array();
$n = count($test_data);
for($i = 0; $i < $n; ++$i){
    $yaxis[$i] = $test_data[$i]['total_cpu'];
}

$xaxis = array();
$n = count($test_data2);
for($i = 0; $i < $n; ++$i){
    $xaxis[$i] = $test_data2[$i]['DATE_FORMAT(timestamp, '%H:%i:%s)'];
}

```

Figure 42. Initializing the variables for graph generation

The process can be seen in Figure 42. When the user has selected the timeframe it is passed to the variables `$sql_sumofprocesses` and `$sql_hour` as `$time1`. The queries are then executed and the data is being stored as an associative array to variables `test_data` and `test_data2`. From the variables appropriate columns are selected and since in this case the total CPU on y-axis and time on x-axis is needed, the corresponding columns are selected. Each result is now stored to an array in their own indexes.

These arrays are then initialized as a session variable which means that they can be used in the php-script to generate the plot (see Figure 43).

```
session_start();
$_SESSION['y'] = $yaxis;
$_SESSION['x'] = $xaxis;

echo 'xaxis->SetTickLabels($datax);

$graph->xgrid->SetColor('#E3E3E3');

// Create the first line
$p1 = new LinePlot($datay);
$graph->Add($p1);
$p1->SetColor("#6495ED");
$p1->SetLegend('Line 1');
```

Figure 44. Generating the plot.

If the PHP-script is run from command line with the test data visible in appendix 7, following graph is generated (see Figure 45).

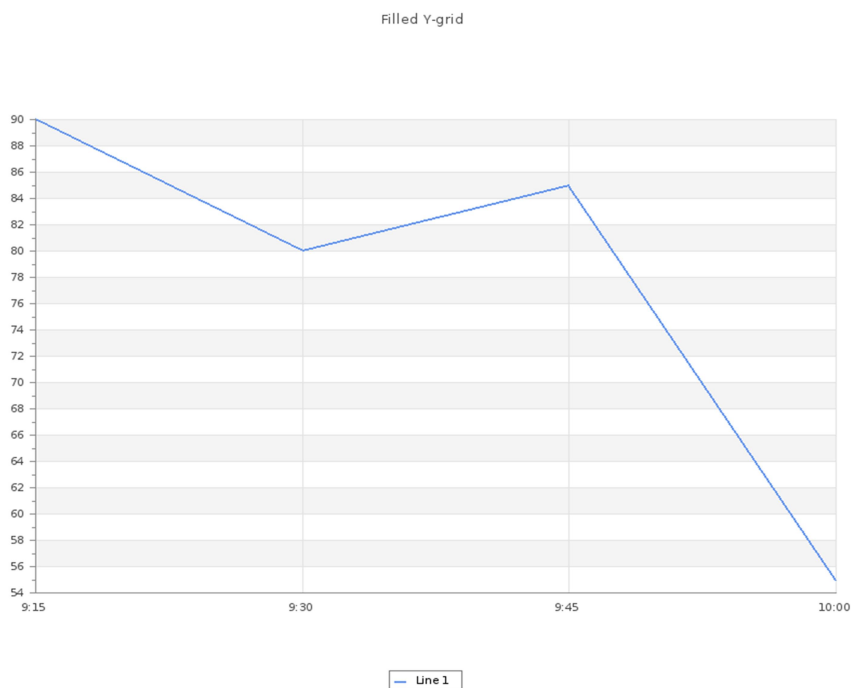


Figure 45. Graph generated with the example data.

The complete php-script can be seen in Appendix 7.

## 6 Discussion and Conclusions

Even though during the development process there were multiple issues found all of the milestones were achieved. A fully working monitoring system was created. The maximum time the system was kept on was three months. There were no issues found during this time. However, it was considered that the risk of running a system update on the Ubuntu Server might cause an error if the Dropbox client was updated. Also, there is no further knowledge on how this system would behave if the host server were updated. This could cause an issue with some execution policy in Powershell or in the task scheduler.

Also, a more detailed performance analyzation would have been useful. At the time Powershell is running the scripts, it uses a high amount of CPU. This can be seen in the data collected and is fabricating the results (see Figure 46).

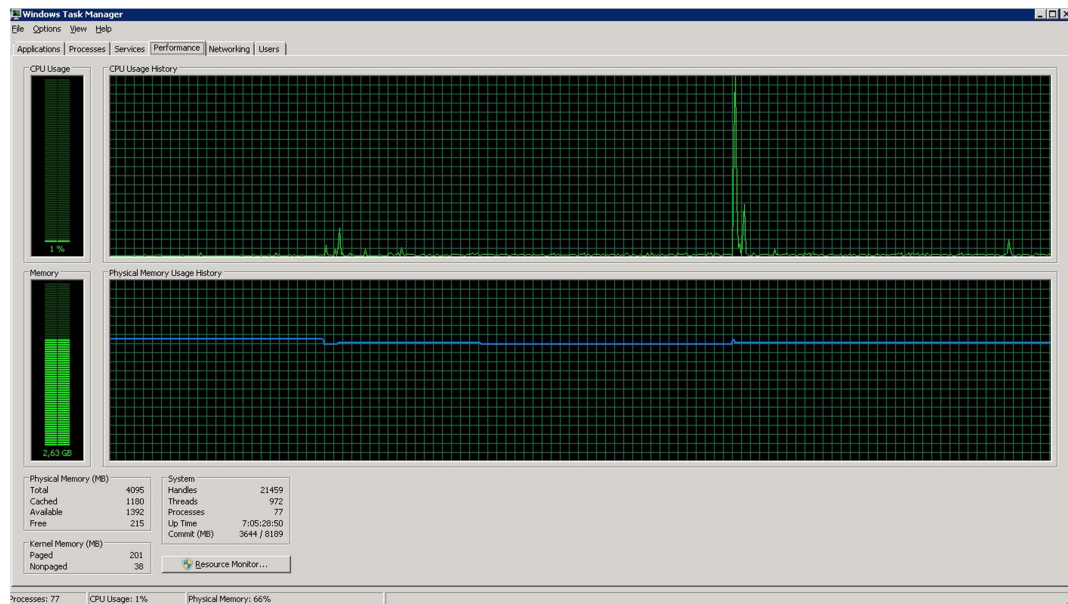


Figure 46. Peak in CPU usage during running of the scripts.

As it was stated in Chapter 4, too many hours of time were wasted when trying to configure the Dropbox client. Even to this day, it is unclear what was the original reason that the latest version of the client was not working. It definitely seemed that this was a global issue.

The first data transfer option was Google Drive, instead of Dropbox client. But during the development process, it was noticed that Google's changing API was hampering the system. This option was then discarded and great amount of time was lost.

## 6.1 Further Development

Overall, the building of the system was a success. However, the amount of time in total spent on the study was significantly underestimated. Lack of experience in web development was one of the reasons which caused unnecessary work and altogether this project was too much work for one person.

This is a prototype. By no means is this a complete system. It still requires a lot of automation, error handling, update handling and smarter backup and storage system. Some of the ideas which have come to mind are as follows:

- Automating the installation of the scripts with only one script
- Different configuration file for credentials and options for choosing the desired functionalities
- Automation the installation of a customer on the Ubuntu server
- Automating the backups and deleting the stored CSV-files after defined time on Ubuntu Server
- Error handling when data connection is not working
- Overall stability testing to monitor the monitoring service

It can be doubted whether the future development is worth of doing since other services such as Zabbix and the latest versions of Nagios and Icinga are very powerful. The point was not to achieve anything greater than these three, but to make something different from a scratch. It safe to say that the goal was reached.



## Sources

- 1 Icinga, 1.1.1. What is Icinga? Article read 10.10.2016  
<https://docs.icinga.com/latest/en/about.html>
- 2 Snover, Jeffery. Q&A-video. Article read 10.10.2016  
<https://community.spiceworks.com/topic/1538108-Powershell-vs-bash-in-windows-10-what-s-the-future-of-Powershell>
- 3 Microsoft Developer Network. How Windows Powershell Works. Article read 10.10.2016 <https://msdn.microsoft.com/en-us/library/ms714658.aspx>
- 4 Microsoft TechNet. Using Windows Powershell to Work with Numbers. Article read 12.10.2016.  
<https://blogs.technet.microsoft.com/heyscriptingguy/2010/08/01/using-windows-Powershell-to-work-with-numbers/>
- 5 Microsoft Developer Network. IPGlobalProperties. Article read 12.10.2016.  
[https://msdn.microsoft.com/en-us/library/system.net.networkinformation.ipglobalproperties\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.net.networkinformation.ipglobalproperties(v=vs.110).aspx)
- 6 Microsoft TechNet. Using the Set-ExecutionPolicy Cmdlet. Article read 28.11.2016 <https://technet.microsoft.com/en-us/library/ee176961.aspx>
- 7 Ask Ubuntu. How to install Dropbox on Ubuntu 14 Server [GUI-less]? Article read: 4.5.2016. <https://askubuntu.com/questions/477425/how-to-install-dropbox-on-ubuntu-14-server-gui-less>
- 8 PHP Manual. Connections. Article read 6.8.2016.  
<http://php.net/manual/en/mysqli.quickstart.connections.php>
- 9 PHP Manual. Mysqli\_real\_connect. Article read 13.12.2016.  
<http://php.net/manual/en/mysqli.real-connect.php>

## Powershell scripts

```

$customer = "etra"
$date = Get-Date -format "d_M_yyyy-HH_mm"
$csvdir = "C:\Users\canteradmin\Dropbox\"

$prop_disks = "ON"
$prop_connections = "ON"
$prop_processes = "ON"
$prop_systemtime = "ON"
$prop_backups = "ON"
$prop_integrations = "ON"
$prop_jboss = "ON"
$prop_changed = "ON"

if ($prop_disks -eq "ON"){
try{
    $opt = "disks"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    Get-WmiObject Win32_logicaldisk -Filter "DeviceID='C:'" |
        Select-Object -
Property DeviceID,
@{n='Size (GB)';e={[Math]::Round((($_.size / 1GB),2)}},
@{n='Free
Space (GB)';e={[Math]::Round((($_.freespace / 1GB),2)}},
@{n='Free(%)';e={[Math]::Round(((($_.freespace / 1GB)*100 / ($_.size /
1GB)),2)}} |
        Export-Csv $outfile -
NoTypeInformation
    }
    catch{
        Write-Error "Failed to get disk data from Etra Test
server"
    }
}

if ($prop_connections -eq "ON"){
try {
    $opt = "connections"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $TCPProperties =
[System.Net.NetworkInformation.IPGlobalProperties]::GetIPGlobalPropert
ies()
    $Connections = $TCPProperties.GetActiveTcpConnections()
    $count = 0
    $localcon = 0
    $ArrList = [System.Collections.ArrayList]@()
    foreach($Connection in $Connections | where {$_.LocalEndPoint.Port
-eq 1098}) {
        $count++ | out-null
        if($Connection.LocalEndPoint.AddressFamily -eq "InterNetwork"
) { $IPType = "IPv4" } else { $IPType = "IPv6" }
        $OutputObj = New-Object -TypeName PSObject
        $OutputObj | Add-Member -MemberType NoteProperty
LocalAddress($Connection.LocalEndPoint.Address) | out-null
    }
}
}

```

```

        $OutputObj | Add-Member -MemberType NoteProperty
LocalPort ($Connection.LocalEndPoint.Port) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
ConnectedAddress ($Connection.RemoteEndPoint.Address) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
RemotePort ($Connection.RemoteEndPoint.Port) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
State ($Connection.State) | out-null
        $OutputObj | Add-Member -MemberType NoteProperty
IPV4Or6 ($IPType) | out-null
        $ArrList.Add($OutputObj) | out-null
        if ($Connection.RemoteEndPoint.Address -eq
$Connection.LocalEndPoint.Address){
            $localcon++
        }
    }

    $result = $Arrlist | select LocalAddress, ConnectedAddress |
Export-Csv $outfile -NoTypeInfoation

} catch {
    Write-Error "Failed to get active connections. $_"
}

if($prop_processes -eq "ON"){
try{
    $opt = "processes"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $CPUPercent = @{n="CPUPercent";e={$TotalSec = (New-TimeSpan -
Start $_.StartTime).TotalSeconds
[Math]::Round(($_.CPU * 100 / $TotalSec), 2)}}

    $proc = Get-Process | Select-Object
Name,@{n="Virtualmemory(GB)";e={[Math]::Round(($_ .VM / 1GB),
2)}},@{n="Workingmemory(GB)";e= {[Math]::Round(($_ .WS / 1GB),
2)}},$CPUPercent, Description | Sort CPUPercent -Descending
    $proc | Export-Csv $outfile -NoTypeInfoation
    $opt = "sumofprocesses"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'

    $sumofcpu = $proc | measure-object CPUPercent -Sum
    $sumofmem = $proc | measure-object "Workingmemory(GB)" -Sum
    $customtable = New-Object -TypeName PSObject
    $customtable | Add-Member -MemberType NoteProperty To-
talCPU($sumofcpu.sum) | out-null
    $customtable | Add-Member -MemberType NoteProperty To-
talMemory($sumofmem.sum) | out-null
    $customtable | Export-Csv $outfile -NoTypeInfoation
    }
    catch{
        Write-Error "Failed to get processes on Etra Test
server"
    }
}

if($prop_systemtime -eq "ON"){
try{
    $opt = "systemtime"

```

```

$outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
$system = Get-WmiObject Win32_operatingsystem
$sup = (Get-Date) - ($system.ConvertToDateTime($system.lastbootuptime))
$boottime = $system.ConvertToDateTime($system.lastbootuptime)
$customsystem = New-Object -TypeName PSObject
$customsystem | Add-Member -MemberType NoteProperty Up-
time("$($sup.Days) days, $($sup.Hours) hours, $($sup.Minutes) minutes")
$customsystem | Add-Member -MemberType NoteProperty
BootTime($boottime)
$customsystem | Export-Csv $outfile -NoTypeInformation
}
catch{
    Write-Error "Failed to get systemtime on Etra Test
server"
}
}

if($prop_backups -eq "ON"){
try{

    $opt = "backups"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $limit = (Get-Date).AddDays(-1)
    $results = Get-Childitem -Path C:\TESTDB2BACKUP -Recurse -
Force | where-object {$_.CreationTime -gt $limit }
    if ($results){
        $backup = $True
    }else{
        $backup = $False
    }

    $amount = (Get-Childitem -Path C:\TESTDB2BACKUP -Recurse -
Force | Measure-Object).Count

    $customback = New-Object -TypeName PSObject
    $customback | Add-Member -MemberType NoteProperty back-
up_name($results)
    $customback | Add-Member -MemberType NoteProperty back-
up_found($backup)
    $customback | Add-Member -MemberType NoteProperty to-
tal_number_of_backups($amount)
    $customback | Export-Csv $outfile -NoTypeInformation
}
catch{
    Write-Error "Failed to get backups on Etra Test serv-
er"
}
}

if($prop_integrations -eq "ON"){
try{

    $opt = "integrations"
    $errors = @("Finished with errors", "error", "Error")
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $limit = (Get-Date).AddDays(-1)
    $integrations = [System.Collections.ArrayList]@()

```

```

$logfile = "C:\integrations\etra_erp2aams\data2aams\logs"
$check = Get-Childitem -Path $logfile | Where-Object
{$_ .Lastwritetime -gt $limit -and $_.extension -eq ".log"}
  if ($check) {
    $status = $check | foreach-object {if (select-string -
pattern $errors "$logfile\$_") {
      $error = $True
    }else{
      $error = $False
    }
    $customint = New-Object -TypeName PSObject
    $customint | Add-Member -MemberType NoteProp-
erty log_name($_.name)
    $customint | Add-Member -MemberType NoteProp-
erty type_name("data2aams")
    $customint | Add-Member -MemberType NoteProp-
erty error_found($error)
    $customint | Add-Member -MemberType NoteProp-
erty write_time($_.Lastwritetime)
    $integrations.Add($customint) | out-null
  }
  }else{
    $customint = New-Object -TypeName PSObject
    $customint | Add-Member -MemberType NoteProp-
erty log_name("NOT FOUND")
    $customint | Add-Member -MemberType NoteProp-
erty type_name("data2aams")
    $customint | Add-Member -MemberType NoteProp-
erty error_found("NOT FOUND")
    $customint | Add-Member -MemberType NoteProp-
erty write_time("NOT FOUND")
    $integrations.Add($customint) | out-null
  }

$logfile = "C:\integrations\etra_erp2aams\erp2xml"
$check = Get-Childitem -Path $logfile | Where-Object
{$_ .Lastwritetime -gt $limit -and $_.extension -eq ".log"}
  if ($check) {
    $status = $check | foreach-object {if (select-string -
pattern $errors "$logfile\$_") {
      $error = $True
    }else{
      $error = $False
    }
    $customint = New-Object -TypeName PSObject
    $customint | Add-Member -MemberType NoteProp-
erty log_name($_.name)
    $customint | Add-Member -MemberType NoteProp-
erty type_name("erp2xml")
    $customint | Add-Member -MemberType NoteProp-
erty error_found($error)
    $customint | Add-Member -MemberType NoteProp-
erty write_time($_.Lastwritetime)
    $integrations.Add($customint) | out-null
  }
  }else{
    $customint = New-Object -TypeName PSObject
    $customint | Add-Member -MemberType NoteProp-
erty log_name("NOT FOUND")

```

```

        $customint | Add-Member -MemberType NoteProp-
erty type_name("erp2xml")
        $customint | Add-Member -MemberType NoteProp-
erty error_found("NOT FOUND")
        $customint | Add-Member -MemberType NoteProp-
erty write_time("NOT FOUND")
        $integrations.Add($customint) | out-null
    }

    $logfile = "C:\integrations\etra_erp2aams\xml2aams\logs"
    $check = Get-Childitem -Path $logfile | Where-Object
{$_ .Lastwritetime -gt $limit -and $_.extension -eq ".log"}
    if ($check){
        $status = $check | foreach-object {if (select-string -
pattern $errors "$logfile\$_"){
            $error = $True
        }else{
            $error = $False
        }
        $customint = New-Object -TypeName PSObject
        $customint | Add-Member -MemberType NoteProp-
erty log_name($_.name)
        $customint | Add-Member -MemberType NoteProp-
erty type_name("xml2aams")
        $customint | Add-Member -MemberType NoteProp-
erty error_found($error)
        $customint | Add-Member -MemberType NoteProp-
erty write_time($_.Lastwritetime)
        $integrations.Add($customint) | out-null
    }
    }else{
        $customint = New-Object -TypeName PSObject
        $customint | Add-Member -MemberType NoteProp-
erty log_name("NOT FOUND")
        $customint | Add-Member -MemberType NoteProp-
erty type_name("xml2aams")
        $customint | Add-Member -MemberType NoteProp-
erty error_found("NOT FOUND")
        $customint | Add-Member -MemberType NoteProp-
erty write_time("NOT FOUND")
        $integrations.Add($customint) | out-null
    }
    $integrations | Export-Csv $outfile -
NoTypeInfoation
    $integrations | Out-File
C:\PowershellScripts\test.csv

}
catch{
    Write-Error "Failed to get backups on Etra Test server"
}

if ($prop_jboss -eq "ON"){
try{
    $opt = "adeonajboss"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    $status = Get-Service "Adeona JBoss" | select name,status
    $status | Export-Csv $outfile -NoTypeInfoation
}
}

```

```

    }
catch{
    Write-Error "Failed to get status of Adeona JBoss on Etra Test
server"
}
}

if($prop_changed -eq "ON"){
try{
    $opt = "changed_processes"
    $outfile = $csvdir + $customer + '_' + $opt + '_' + $date +
'.csv'
    Get-Process | Select-Object Name | Export-Csv
C:\PowerMon\files\process_comparator.csv
    $a = Get-Content C:\PowerMon\files\initial_process_list.csv
    $b = Get-Content C:\PowerMon\files\process_comparator.csv
    $arr = [System.Collections.ArrayList]@()
    $rows = Compare-Object $a $b

    if($rows){
        foreach ($row in $rows){
            if($row.SideIndicator -eq "=>"){
                $state = "NEW"
            }
            if($row.SideIndicator -eq "<="){
                $state = "OLD"
            }

            $changed = New-Object -TypeName PSObject
            $changed | Add-Member -MemberType NoteProperty
Name($row.InputObject)
            $changed | Add-Member -MemberType NoteProperty
State($state)
            $arr.Add($changed) | Out-Null
        }
        $arr | Export-Csv $outfile -NoTypeInfoation
        Remove-Item
C:\PowerMon\files\initial_process_list.csv -force
        Rename-Item
C:\PowerMon\files\process_comparator.csv -NewName ini-
tial_process_list.csv
    }
}
catch{
    Write-Error "Failed to get changed processes data"
}
}
}

```

## Script for creating the database tables

```
#!/usr/bin/php

<?php
    $server = "localhost";
    $username = "root";
    $password = "*****";
    $db = "etra_power";

    $conn = new mysqli($server, $username, $password, $db);
    if ($conn->connect_error){
        die("Connection failed: " . $conn->connect_error);
    }

    $sql_disks = "CREATE TABLE IF NOT EXISTS etra_disks (
        id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        device_id VARCHAR(5),
        size DOUBLE(10,2),
        free_space DOUBLE(10,2),
        free_percent DOUBLE(10,2),
        timestamp DATETIME
    )";

    $sql_processes = "CREATE TABLE IF NOT EXISTS etra_processes (
        id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR (50),
        virtualmemory DOUBLE(10,2),
        workingmemory DOUBLE(10,2),
        cpupercent DOUBLE(10,2),
        description VARCHAR(50),
        timestamp DATETIME
    )";

    $sql_connections = "CREATE TABLE IF NOT EXISTS et-
ra_connections (
        id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        localaddress VARCHAR(20),
        connectedaddress VARCHAR(20),
        timestamp DATETIME
    )";

    $sql_adeonajboss = "CREATE TABLE IF NOT EXISTS et-
ra_adeonajboss (
        id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(40),
        status VARCHAR(10),
        timestamp DATETIME
    )";

    $sql_backups = "CREATE TABLE IF NOT EXISTS etra_backups (
        id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        backup_name VARCHAR(50),
        backup_found VARCHAR(10),
        backups_amount INT,
        timestamp DATETIME
    )";
```



```

    $sql_integrations = "CREATE TABLE IF NOT EXISTS et-
ra_integrations (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    log_name VARCHAR(50),
    type_name VARCHAR(50),
    error_found VARCHAR(50),
    write_time DATETIME,
    timestamp DATETIME
    )";

    $sql_sumofprocesses = "CREATE TABLE IF NOT EXISTS et-
ra_sumofprocesses (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    total_cpu DOUBLE(6,2),
    total_memory DOUBLE(6,2),
    timestamp DATETIME
    )";

    $sql_changedprocesses = "CREATE TABLE IF NOT EXISTS et-
ra_changedprocesses (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    timestamp DATETIME
    )";

    $sql_systemtime = "CREATE TABLE IF NOT EXISTS etra_systemtime
(
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    uptime VARCHAR(50),
    boottime VARCHAR(50),
    timestamp DATETIME
    )";

    if ($conn->query($sql_disks) === TRUE){
        echo "Table etra_disks created successfully\n";
    }else{
        echo "Error creating table etra_disks:\n " . $conn-
>error;
    }

    if ($conn->query($sql_processes) === TRUE){
        echo "Table etra_processes created successfully\n";
    }else{
        echo "Error creating table etra_processes:\n " .
$conn->error;
    }

    if ($conn->query($sql_connections) === TRUE){
        echo "Table etra_connections created successfully\n";
    }else{
        echo "Error creating table etra_connections:\n " .
$conn->error;
    }

    if ($conn->query($sql_adeonajboss) === TRUE){
        echo "Table etra_adeonajboss created successfully\n";
    }else{
        echo "Error creating table etra_adeonajboss:\n " .
$conn->error;
    }

```

```
        if ($conn->query($sql_backups) === TRUE){
            echo "Table etra_backups created successfully\n";
        }else{
            echo "Error creating table etra_etra_backups:\n " .
$conn->error;
        }

        if ($conn->query($sql_integrations) === TRUE){
            echo "Table etra_integrations created successfully\n";
        }else{
            echo "Error creating table etra_integrations:\n " .
$conn->error;
        }

        if ($conn->query($sql_sumofprocesses) === TRUE){
            echo "Table etra_sumofprocesses created successful-
ly\n";
        }else{
            echo "Error creating table etra_sumofprocesses:\n " .
$conn->error;
        }

        if ($conn->query($sql_changedprocesses) === TRUE){
            echo "Table etra_changedprocesses created successful-
ly\n";
        }else{
            echo "Error creating table etra_changedprocesses:\n "
. $conn->error;
        }

        if ($conn->query($sql_systemtime) === TRUE){
            echo "Table etra_systemtime created successfully\n";
        }else{
            echo "Error creating table etra_systemtime:\n " .
$conn->error;
        }

        $conn->close();
?>
```

## Script for importing the CSV files

```
#!/usr/bin/php
```

```
<?php
```

```
error_reporting( error_reporting() & ~E_NOTICE );

$file_disks = '/home/psdownloads/etra_disks *';
$file_processes = '/home/psdownloads/etra_processes *';
$file_connections = '/home/psdownloads/etra_connections *';
$file_sumproc = '/home/psdownloads/etra_sumofprocesses *';
$file_systemtime = '/home/psdownloads/etra_systemtime *';
$file_backups = '/home/psdownloads/etra_backups *';
$file_integrations = '/home/psdownloads/etra_integrations *';
$file_jboss = '/home/psdownloads/etra_adeonajboss *';
$file_changed = '/home/psdownloads/etra_changed_processes *';

foreach (glob($file_disks) as $etra_diskfile){
    echo "File $etra_diskfile found!\n";
}

foreach (glob($file_processes) as $etra_procfile){
    echo "File $etra_procfile found!\n";
}

foreach (glob($file_connections) as $etra_connfile){
    echo "File $etra_connfile found!\n";
}

foreach (glob($file_sumproc) as $etra_sumprocfile){
    echo "File $etra_sumprocfile found!\n";
}

foreach (glob($file_systemtime) as $etra_systimefile){
    echo "File $etra_systimefile found!\n";
}

foreach (glob($file_backups) as $etra_backupfile){
    echo "File $etra_backupfile found!\n";
}

foreach (glob($file_integrations) as $etra_integrationsfile){
    echo "File $etra_integrationsfile found!\n";
}

foreach (glob($file_jboss) as $etra_jbossfile){
    echo "File $etra_jbossfile found!\n";
}

foreach (glob($file_changed) as $etra_changedfile){
    echo "File $etra_changedfile found!\n";
}
```

```

$user = 'root';
$pass = '*****';
$db = 'etra_power';
$host = 'localhost';

$mysqli = mysqli_init();

mysqli_options($mysqli, MYSQLI_OPT_LOCAL_INFILE, true);
#Initialize the database connection
mysqli_real_connect($mysqli,$host, $user, $pass,$db)
    or die ('<P>ERROR connecting to $db</P>');

#Initialize the MYSQL queries
$query_disks = "LOAD DATA LOCAL INFILE '$etra_diskfile'
    INTO TABLE etra_disks
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (device id,size,free space,free_percent)
    SET timestamp=NOW()";

$query_processes = "LOAD DATA LOCAL INFILE '$etra_procfile'
    INTO TABLE etra_processes
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (name,virtualmemory,workingmemory,cpupercent,description)
    SET timestamp=NOW()";

$query_connections = "LOAD DATA LOCAL INFILE '$etra_connfile'
    INTO TABLE etra connections
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (localaddress,connectedaddress)
    SET timestamp=NOW()";

$query_jboss = "LOAD DATA LOCAL INFILE '$etra_jbossfile'
    INTO TABLE etra_adeonajboss
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (name,status)
    SET timestamp=NOW()";

$query_sumofproc = "LOAD DATA LOCAL INFILE
'$etra_sumprocfile'
    INTO TABLE etra sumofprocesses
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 LINES
    (total_cpu,total_memory)
    SET timestamp=NOW()";

$query_systime = "LOAD DATA LOCAL INFILE '$etra_systimefile'
    INTO TABLE etra_systime
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
    LINES TERMINATED BY '\r\n'

```

```

IGNORE 1 LINES
(uptime,boottime)
SET timestamp=NOW()";

$query_backups = "LOAD DATA LOCAL INFILE '$etra_backupfile'
INTO TABLE etra_backups
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(backup_name,backup_found,backups_amount)
SET timestamp=NOW()";

$query_integrations = "LOAD DATA LOCAL INFILE
'$etra_integrationsfile'
INTO TABLE etra_integrations
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(log_name,type_name,error_found,write_time)
SET timestamp=NOW()";

$query_changed = "LOAD DATA LOCAL INFILE '$etra_changedfile'
INTO TABLE etra_changedprocesses
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(name,state)
SET timestamp=NOW()";

#Execute queries
if ($mysql->query($query_disks) === TRUE){
    echo "Data imported succesfully\n";
}else{
    echo "Error importing data: \n" . $mysql->
>error;
}

if ($mysql->query($query_processes) === TRUE){
    echo "Data imported succesfully\n";
}else{
    echo "Error importing data: \n" . $mysql->
>error;
}

if ($mysql->query($query_connections) === TRUE){
    echo "Data imported succesfully\n";
}else{
    echo "Error importing data: \n" . $mysql->
>error;
}

if ($mysql->query($query_changed) === TRUE){
    echo "Data imported succesfully\n";
}else{
    echo "Error importing data: \n" . $mysql->
>error;
}

```

```
        if ($mysqli->query($query_integrations) === TRUE) {
            echo "Data imported succesfully\n";
        }else{
            echo "Error importing data: \n" . $mysqli->
>error;
        }

        if ($mysqli->query($query_backups) === TRUE){
            echo "Data imported succesfully\n";
        }else{
            echo "Error importing data: \n" . $mysqli->
>error;
        }

        if ($mysqli->query($query_systime) === TRUE){
            echo "Data imported succesfully\n";
        }else{
            echo "Error importing data: \n" . $mysqli->
>error;
        }

        if ($mysqli->query($query_sumofproc) === TRUE){
            echo "Data imported succesfully\n";
        }else{
            echo "Error importing data: \n" . $mysqli->
>error;
        }

        if ($mysqli->query($query_jboss) === TRUE){
            echo "Data imported succesfully\n";
        }else{
            echo "Error importing data: \n" . $mysqli->
>error;
        }

        $mysqli->close();

        $files = scandir("/home/psdownloads");
        $source = "/home/psdownloads/";
        $destination = "/home/psarchive/";
        foreach ($files as $file) {
            if (in_array($file, array(".", ".."))) continue;
            if (copy($source.$file, $destination.$file)) {
                $delete[] = $source.$file;
            }
        }
        foreach ($delete as $file) {
            unlink($file);
        }

?>
```

## Source code for the index page

```

<!DOCTYPE HTML>
<html>

<head>
  <title>PowerMon</title>
  <meta name="description" content="website description" />
  <meta name="keywords" content="website keywords, website keywords"
/>
  <meta http-equiv="content-type" content="text/html; charset=windows-
1252" />
  <link rel="stylesheet" type="text/css" href="style/style.css" ti-
tle="style" />
</head>

<body>
  <div id="main">
    <div id="header">
      <div id="logo">
        <div id="logo_text">
          <!-- class="logo_colour", allows you to change the colour of
the text -->
          <br>
          <br>
          <br>
          <br>
          <br>
          <h2>Cost-efficient monitoring services</h2>
        </div>
      </div>
      <div id="menubar">
        <ul id="menu">

          <li class="selected"><a href="index.html">Home</a></li>
          <li><a href="datatables.php">Data tables</a></li>
          <li><a href="graphs.php">Graphs</a></li>
        </ul>
      </div>
    </div>
    <div id="site_content">
      <div id="content">

        <h1>Welcome to the Powershell Monitoring System webpage</h1>
        <p>The purpose of this project is to monitor servers (at the
moment Windows Servers) without any cost of third party actions. In
smaller companies monitoring is usually impl  emented with a service
provider which results in extra costs. By extra costs I mean support
requests to ISP to i.e open specific ports or connections.</p>

        <p>With this powerful yet simple product ISPs and
these extra costs can be removed from this chain since all the re-
quests are made locally on the server and then sen  t via Google
Cloud. MySQL database fetches the data from the cloud and the data is
then viewed in web.</p>

```

```
<p>Please not that this is a work in progress and is only a  
prototype. Since there is a certain limit of hours I can use.....</p>
```

```
    </ul>  
  </div>  
</div>  
<div id="content_footer"></div>  
<div id="footer">  
  This monitoring system is provided by Aki Riisiö  
</div>  
</div>  
</body>  
</html>
```



## Source code for data tables page

```

<?php
    error_reporting(E_ALL);
    session_set_cookie_params(60);
    session_start();
    #phpinfo();

?><!DOCTYPE HTML>
<html>

<head>
    <title>PowerMon - Datatables</title>
    <meta name="description" content="website description" />
    <meta name="keywords" content="website keywords, website keywords"
/>
    <meta http-equiv="content-type" content="text/html; charset=windows-
1252" />
    <link rel="stylesheet" type="text/css" href="style/style.css" ti-
tle="style" />
</head>

<body>
    <div id="main">
        <div id="header">
            <div id="logo">
                <div id="logo_text">
                    <br>
                    <br>
                    <br>
                    <br>
                    <br>
                    <h2>Cost-efficient monitoring services</h2>
                </div>
            </div>
            <div id="menubar">
                <ul id="menu">
                    <li><a href="index.html">Home</a></li>
                    <li class="selected"><a href="datatables.php">Data ta-
bles</a></li>
                    <li><a href="graphs.php">Graphs</a></li>
                </ul>
            </div>
            <div id="content_header"></div>
            <div id="site_content_v2">
                <div id="content_v2">

<?php
    $servername = "localhost";
    $username = "root";
    $password = "*****";

```

```

$dbname = "etra_power";

?>

<h1>Disk information</h1>
<?php

function createDropdown($option) {

    $list = array(
        array("1", "Now"),
        array("2", "30 minutes"),
        array("3", "1 hour"),
        array("4", "2 hours"),
        array("5", "4 hours"),
        array("6", "8 hours"),
        array("7", "12 hours"),
        array("8", "1 day"),
        array("9", "2 days"),
        array("10", "4 days"),
        array("11", "1 week"),
        array("12", "2 weeks"),
        array("13", "1 month"),
        ar-
ray("14", "--disabled--")
    );

    foreach ($list as &$value) {
        echo "<option ";
        if($value[0] == $option){
            echo 'selected="selected"';
        }
        echo " value=";
        echo $value[0];
        echo ">";
        echo $value[1];
        echo "</option>";
    }

}

function create_form($timeframe) {

    $x1 = $_POST[$timeframe];
    if (!isset($x1)) {
        if(isset($_SESSION[$timeframe])) {
            $x1 = $_SESSION[$timeframe];
        }
    }
    $_SESSION[$timeframe] = $x1;

?>

<form action="" method="POST">
    <div class="form_settings">
        <p><span><b>Select
timeframe</b></span><br><select id="<?php echo $timeframe; ?>"
name="<?php echo $timeframe; ?>"></p>

```

```

        <?php createDropdown($x1); ?>
        <p style="padding-right:
15px"><span>&nbsp;</span><input class="submit" type="submit"
name="name" value="GO" /></p>

        <?php

                if($x1 == 1){ $time1 = 'NOW() - INTERVAL 15
MINUTE'; };
                if($x1 == 2){ $time1 = 'NOW() - INTERVAL 30
MINUTE'; };
                if($x1 == 3){ $time1 = "NOW() - INTERVAL 1
HOUR"; };
                if($x1 == 4){ $time1 = "NOW() - INTERVAL 2
HOUR"; };
                if($x1 == 5){ $time1 = "NOW() - INTERVAL 4
HOUR"; };
                if($x1 == 6){ $time1 = "NOW() - INTERVAL 8
HOUR"; };
                if($x1 == 7){ $time1 = "NOW() - INTERVAL 12
HOUR"; };
                if($x1 == 8){ $time1 = "NOW() - INTERVAL 1
DAY"; };
                if($x1 == 9){ $time1 = "NOW() - INTERVAL 2
DAY"; };
                if($x1 == 10){ $time1 = "NOW() - INTERVAL 4
DAY"; };
                if($x1 == 11){ $time1 = "NOW() - INTERVAL 7
DAY"; };
                if($x1 == 12){ $time1 = "NOW() - INTERVAL 14
DAY"; };
                if($x1 == 13){ $time1 = 'NOW() - INTERVAL 31
DAY;'; };
                if($x1 == 14){ $time1 = 'DISABLED'; };

        ?>
    </div>
</form>

    <?php
    return $time1;
    }

    $time1 = create_form('timeframe1');

    // Create connection
    $conn = new mysqli($servername, $username, $password,
$dbname);
    // Check connection
    if ($conn->connect_error){
        die("Connection failed: " . $conn-
>connect_error);
    }

    $sql_disk = "SELECT de-
vice_id,size,free_space,free_percent,timestamp FROM etra_disks WHERE
timestamp >= $time1;";
    $result = $conn->query($sql_disk);

```

```

# $sql_connections

if($time1 == 'DISABLED'){
    echo "<b>Option disabled</b>";
} else{
    echo "<table style='width:100%; border-spacing:0;'>
        <tr>
            <th>Device ID</th>
            <th>Disk size</th>
            <th>Free space</th>
            <th>Free space (%)</th>
            <th>Timestamp</th>
        </tr>";

    if ($result->num_rows > 0){
        // output data of each row
        while($row = $result->fetch_assoc()){
            echo "<tr>";
            echo "<td>" . $row['device_id'] .
"</td>";
            echo "<td>" . $row['size'] . "</td>";
            echo "<td>" . $row['free_space'] .
"</td>";
            echo "<td>" . $row['free_percent'] .
"</td>";
            echo "<td>" . $row['timestamp'] .
"</td>";
        }
    } else{
        echo "Could not find data in the given
timeframe!";
    }

    echo "</table>";
}

$conn->close();

?>

<h1>Systemtime information</h1>

<?php

$time2 = create_form('timeframe2');

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
// Check connection
if ($conn->connect_error){
    die("Connection failed: " . $conn-
>connect_error);
}

$sql_systemtime = "SELECT uptime,boottime,timestamp
FROM etra_systemtime WHERE timestamp >= $time2";
$result = $conn->query($sql_systemtime);

# $sql_connections

```

```

        if($time2 == 'DISABLED'){
            echo "<b>Option disabled</b>";
        }else{

            echo "<table style='width:100%; border-spacing:0;'>
                <tr>
                    <th>Uptime</th>
                    <th>Last boottime</th>
                    <th>Timestamp</th>
                </tr>";

            if ($result->num_rows > 0){
                // output data of each row
                while($row = $result->fetch_assoc()){
                    echo "<tr>";
                    echo "<td>" . $row['uptime'] .
"</td>";
                    echo "<td>" . $row['boottime'] .
"</td>";
                    echo
"<td>" . $row['timestamp'] . "</td>";
                }
                }else{
                    echo "Could not find data in the given
timeframe!";
                }

            echo "</table>";
        }
        $conn->close();
    ?>

```

<h1>Overall process information</h1>

<?php

```

    $time3 = create_form('timeframe3');

    $servername = "localhost";
    $username = "root";
    $password = "*****";
    $dbname = "etra_power";

    // Create connection
    $conn = new mysqli($servername, $username, $password,
$dbname);
    // Check connection
    if ($conn->connect_error){
        die("Connection failed: " . $conn-
>connect_error);
    }

    $sql_sumofprocesses = "SELECT to-
tal_cpu,total_memory,timestamp FROM etra_sumofprocesses WHERE
timestamp >= $time3";

```

```

        $result = $conn->query($sql_sumofprocesses);

        # $sql_connections

        if($time3 == 'DISABLED'){
            echo "<b>Option disabled</b>";
        }else{

            echo "<table style='width:100%; border-spacing:0;'>
                <tr>
                <th>Total CPU-load (%)</th>
                <th>Total memory usage</th>
                <th>Timestamp</th>
                </tr>";

            if ($result->num_rows > 0){
                // output data of each row
                while($row = $result->fetch_assoc()){
                    echo "<tr>";
                    echo "<td>" . $row['total_cpu'] .
"</td>";
                    echo "<td>" . $row['total_memory'] .
"</td>";
                    echo
"<td>" . $row['timestamp'] . "</td>";
                }
            }else{
                echo "Could not find data in the given
timeframe!";
            }
            echo "</table>";
        }
        $conn->close();
    ?>

    <h1>Process information</h1>

    <?php

        $time4 = create_form('timeframe4');

        // Create connection
        $conn = new mysqli($servername, $username, $password,
$dbname);

        // Check connection
        if ($conn->connect_error){
            die("Connection failed: " . $conn-
>connect_error);
        }

        $sql_processes = "SELECT
name,virtualmemory,workingmemory,cpupercent,timestamp FROM et-
ra_processes WHERE timestamp >= $time4;";
        $result = $conn->query($sql_processes);

        # $sql_connections

        if($time4 == 'DISABLED'){

```

```

        echo "<b>Option disabled</b>";
    }else{

        echo "<table style='width:100%; border-spacing:0;'>
            <tr>
                <th>Name</th>
                <th>Virtual memory</th>
                <th>Working memory</th>
                <th>CPU (%)</th>
                <th>Timestamp</th>
            </tr>";

        if ($result->num_rows > 0){
            // output data of each row
            while($row = $result->fetch_assoc()){
                echo "<tr>";
                echo "<td>" . $row['name'] . "</td>";
                echo "<td>" . $row['virtualmemory'] .
"</td>";
                echo "<td>" . $row['workingmemory'] .
"</td>";
                echo "<td>" . $row['cpupercent'] .
"</td>";
                echo
"<td>" . $row['timestamp'] . "</td>";
                echo "<td>" . $row['description'] .
"</td>";
            }
        }else{
            echo "Could not find data in the given
timeframe!";
        }

        echo "</table>";
    }
    $conn->close();
?>

<h1>Changed process information</h1>
<?php

$time5 = create_form('timeframe5');

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
// Check connection
if ($conn->connect_error){
    die("Connection failed: " . $conn-
>connect_error);
}

$sql_changedproc = "SELECT name,state,timestamp FROM
etra_changedprocesses WHERE timestamp >= $time5";
$result = $conn->query($sql_changedproc);

# $sql_connections

```

```

        if($time5 == 'DISABLED'){
            echo "<b>Option disabled</b>";
        }else{

            echo "<table style='width:100%; border-spacing:0;'>
                <tr>
                <th>Name</th>
                <th>Old / New</th>
                <th>Timestamp</th>
                </tr>";

            if ($result->num_rows > 0){
                // output data of each row
                while($row = $result->fetch_assoc()){
                    echo "<tr>";
                    echo "<td>" . $row['name'] . "</td>";
                    echo "<td>" . $row['state'] . "</td>";
                    echo "<td>" . $row['timestamp'] .
"</td>";
                }
            }else{
                echo "Could not find data in the given
timeframe!";
            }

            echo "</table>";
        }
        $conn->close();
    ?>

    <h1>Adeona connection information</h1>
    <?php

        $time6 = create_form('timeframe6');

        // Create connection
        $conn = new mysqli($servername, $username, $password,
$dbname);
        // Check connection
        if ($conn->connect_error){
            die("Connection failed: " . $conn-
>connect_error);
        }

        $sql_connections = "SELECT localad-
dress,connectedaddress,timestamp FROM etra_connections WHERE timestamp
>= $time6;";
        $result = $conn->query($sql_connections);

        # $sql_connections

        if($time6 == 'DISABLED'){
            echo "<b>Option disabled</b>";
        }else{

            echo "<table style='width:100%; border-spacing:0;'>
                <tr>
                <th>Local address</th>

```



```

        <th>Connected address</th>
        <th>Timestamp</th>
    </tr>";

    if ($result->num_rows > 0){
        // output data of each row
        while($row = $result->fetch_assoc()){
            echo "<tr>";
            echo "<td>" . $row['localaddress'] .
"</td>";
            echo "<td>" . $row['connectedaddress']
. "</td>";
            echo
"<td>" . $row['timestamp'] . "</td>";
        }
    }else{
        echo "Could not find data in the given
timeframe!";
    }

    echo "</table>";
}
$conn->close();
?>

<h1>Adeona status information</h1>
<?php

$time7 = create_form('timeframe7');

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
// Check connection
if ($conn->connect_error){
    die("Connection failed: " . $conn-
>connect_error);
}

$sql_adeona = "SELECT name,status,timestamp FROM et-
ra_adeonajboss WHERE timestamp >= $time7;";
$result = $conn->query($sql_adeona);

# $sql_connections
if($time7 == 'DISABLED'){
    echo "<b>Option disabled</b>";
}else{

    echo "<table style='width:100%; border-spacing:0;'>
    <tr>
    <th>Name</th>
    <th>Status</th>
    <th>Timestamp</th>
    </tr>";

    if ($result->num_rows > 0){
        // output data of each row

```

```

        while($row = $result->fetch_assoc()){
            echo "<tr>";
            echo "<td>" . $row['name'] . "</td>";
            echo "<td>" . $row['status'] .
"</td>";
                                                                    echo
"<td>" . $row['timestamp'] . "</td>";
        }
    }else{
        echo "Could not find data in the given
timeframe!";
    }

    echo "</table>";
}
$conn->close();
?>

<h1>Backups information</h1>
<?php

$time8 = create_form('timeframe8');

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
// Check connection
if ($conn->connect_error){
    die("Connection failed: " . $conn-
>connect_error);
}

$sql_backups = "SELECT back-
up_name,backup_found,backups_amount,timestamp FROM etra_backups WHERE
timestamp >= $time8;";
$result = $conn->query($sql_backups);

# $sql_connections

if($time8 == 'DISABLED'){
    echo "<b>Option disabled</b>";
}else{

    echo "<table style='width:100%; border-spacing:0;'>
        <tr>
            <th>Name</th>
            <th>Found</th>
            <th>Amount of backups</th>
            <th>Timestamp</th>
        </tr>";

    if ($result->num_rows > 0){
        // output data of each row
        while($row = $result->fetch_assoc()){
            echo "<tr>";
            echo "<td>" . $row['backup_name'] .
"</td>";

```

```

                                echo "<td>" . $row['backup_found'] .
"</td>";
                                echo "<td>" . $row['backups_amount'] .
"</td>";
                                                                    echo
"<td>" . $row['timestamp'] . "</td>";
                                }
                                }else{
                                echo "Could not find data in the given
timeframe!";
                                }

                                echo "</table>";
                                }
                                $conn->close();
                                ?>

                                <h1>Integrations information</h1>
                                <?php

                                $time9 = create_form('timeframe9');

                                // Create connection
                                $conn = new mysqli($servername, $username, $password,
$dbname);
                                // Check connection
                                if ($conn->connect_error){
                                die("Connection failed: " . $conn-
>connect_error);
                                }

                                $sql_integrations = "SELECT
log_name,type_name,error_found,write_time,timestamp FROM et-
ra_integrations WHERE timestamp >= $time9;";
                                $result = $conn->query($sql_integrations);

                                if($time9 == 'DISABLED'){
                                echo "<b>Option disabled</b>";
                                }else{

                                echo "<table style='width:100%; border-spacing:0;'>
                                <tr>
                                <th>Log name</th>
                                <th>Type</th>
                                <th>Errors</th>
                                <th>Write time</th>
                                <th>Timestamp</th>
                                </tr>";

                                if ($result->num_rows > 0){
                                // output data of each row
                                while($row = $result->fetch_assoc()){
                                echo "<tr>";
                                echo "<td>" . $row['log_name'] .
"</td>";
                                echo "<td>" . $row['type_name'] .

```

```
                echo "<td>" . $row['error_found'] .
"</td>";
                echo "<td>" . $row['write_time'] .
"</td>";
                echo
"&td>" . $row['timestamp'] . "</td>";
            }
        }else{
            echo "Could not find data in the given
timeframe!";
        }

        echo "</table>";
    }
    $conn->close();

    session_write_close();

    ?>

</div>

<div id="content_footer"></div>
<div id="footer">
    This monitoring system is provided by Aki Riisiö</div>
</div>
</body>
</html>
```

## Source code for graphs page

```

<!DOCTYPE HTML>
<html>

<?php

    $servername = "localhost";
    $username = "root";
    $password = "*****";
    $dbname = "etra_power";

    ?>

<head>
  <title>colour_blue - another page</title>
  <meta name="description" content="website description" />
  <meta name="keywords" content="website keywords, website keywords"
  />
  <meta http-equiv="content-type" content="text/html; charset=windows-
1252" />
  <link rel="stylesheet" type="text/css" href="style/style.css" ti-
tle="style" />
</head>

<body>
  <div id="main">
    <div id="header">
      <div id="logo">
        <div id="logo_text">
          <!-- class="logo_colour", allows you to change the colour of
the text -->
          <br>
          <br>
          <br>
          <br>
          <h2>Cost-efficient monitoring services</h2>
        </div>
      </div>
      <div id="menubar">
        <ul id="menu">

          <li><a href="index.html">Home</a></li>
          <li><a href="datatables.php">Data tables</a></li>
          <!--<li><a href="files.php">Files</a></li>-->
          <li class="selected"><a href="graphs.php">Graphs</a></li>
          <!--<li><a href="contact.php">Contact Me</a></li>-->
        </ul>
      </div>
    </div>
    <div id="content_header"></div>
    <div id="site_content">
      <div class="sidebar">
        <!-- insert your sidebar items here -->
        <h3>Latest News</h3>

```

```
<h4>The project starts...</h4>
<h5>June 1st, 2015</h5>
```

```
<p>It is time to start the project. Most of the Powershell
scripts have already been made...!</p>
```

```
<h3>Useful Links</h3>
<ul>
  <li><a href="#">link 1</a></li>
</ul>
</div>
<div id="content">
  <!-- insert the page content here -->
```

```
<h1>CPU LOAD GRAPH</h1>
<?php
```

```
function createDropdown($option) {
    $list = array(
        array("1", "1 hour"),
        array("2", "2 hours"),
        array("3", "4 hours"),
        array("4", "8 hours"),
        array("5", "12 hours"),
        array("6", "--disabled--")
    );

    foreach ($list as &$value) {
        echo "<option ";
        if($value[0] == $option){
            echo 'selected="selected"';
        }
        echo " value=";
        echo $value[0];
        echo ">";
        echo $value[1];
        echo "</option>";
    }
}

function create_form($timeframe) {
    $x1 = $_POST[$timeframe];
    if (!isset($x1)) {
        if(isset($_SESSION[$timeframe])) {
            $x1 = $_SESSION[$timeframe];
        }
    }

    $_SESSION[$timeframe] = $x1;

    ?>
    <form action="" method="POST">
        <div class="form_settings">
```

```

                                <p><span><b>Select
timeframe</b></span><br><select id="<?php echo $timeframe; ?>"
name="<?php echo $timeframe; ?>"></p>

                                <?php createDropdown($x1); ?>
                                <p style="padding-right:
15px"><span>&nbsp;</span><input class="submit" type="submit"
name="name" value="GO" /></p>

                                <?php

                                if($x1 == 1){ $time1 = "NOW() - INTERVAL 1
HOUR"; };
                                if($x1 == 2){ $time1 = "NOW() - INTERVAL 2
HOUR"; };
                                if($x1 == 3){ $time1 = "NOW() - INTERVAL 4
HOUR"; };
                                if($x1 == 4){ $time1 = "NOW() - INTERVAL 8
HOUR"; };
                                if($x1 == 5){ $time1 = "NOW() - INTERVAL 12
HOUR"; };
                                if($x1 == 6){ $time1 = 'DISABLED'; };

                                ?>
                                </div>
                                </form>

                                <?php

                                session_start();
                                $_SESSION['time'] = $time1;
                                return $time1;
                                }

                                $time1 = create_form('timeframe1');

                                $conn = new mysqli($servername, $username, $password,
$dbname);
                                // Check connection
                                if ($conn->connect_error){
                                    die("Connection failed: " . $conn->connect_error);
                                }

                                $sql_sumofprocesses = "SELECT total_cpu FROM et-
ra_sumofprocesses WHERE timestamp >= $time1;";
                                $sql_hour = "SELECT DATE_FORMAT(timestamp, '%H:%i:%s') FROM
etra_sumofprocesses WHERE timestamp >= $time1;";
                                $result = $conn->query($sql_sumofprocesses);
                                $test_data = array();
                                if ($result->num_rows > 0){
                                    // output data of each row
                                    while($row = $result->fetch_assoc()){
                                        $test_data[] = $row;
                                    }
                                }else{
                                    echo "Could not find data in the given
timeframe!";
                                }

```

```

$result = $conn->query($sql_hour);
$test_data2 = array();
if ($result->num_rows > 0){
    // output data of each row
    while($row = $result->fetch_assoc()){
        $test_data2[] = $row;
    }
}
else{
    echo "Could not find data in the given
timeframe!";
}

$conn->close();

$yaxis = array();
$n = count($test_data);
for($i = 0; $i < $n; ++$i){
    $yaxis[$i] = $test_data[$i]['total_cpu'];
}

$xaxis = array();
$n = count($test_data2);
for($i = 0; $i < $n; ++$i){
    $xaxis[$i] =
$test_data2[$i]["DATE_FORMAT(timestamp, '%H:%i:%s')"];
}

?>

<?php

//$xaxis = array('9:00','9:15','9:30','9:45');
//$yaxis = array(30,80,95,55);

session_start();
$_SESSION['y'] = $yaxis;
$_SESSION['x'] = $xaxis;

echo '</div>
<div id="footer">This monitoring system is provided by Aki
Riisiö</div>
</div>
</body>
</html>

```



## Source code for generating the graph

```

<?php // content="text/plain; charset=utf-8"

    require_once ('/home/downloads/jpgraph-
3.5.0b1/src/jpgraph.php');
    require_once ('/home/downloads/jpgraph-
3.5.0b1/src/jpgraph_line.php');
    require_once ('/home/downloads/jpgraph-
3.5.0b1/src/jpgraph_date.php');
    require_once ('/home/downloads/jpgraph-
3.5.0b1/src/jpgraph_utils.inc.php');

    session_start();
    $datay = $_SESSION['y'];
    $datax = $_SESSION['x'];

    //We fetched the data stored in the sessions to variables
datay and datax and now we can use them

    // Setup the graph
    $graph = new Graph(800,800);
    $graph->SetScale("textlin");

    $theme_class=new UniversalTheme;

    $graph->SetTheme($theme_class);
    $graph->img->SetAntiAliasing(false);
    $graph->title->Set('Filled Y-grid');
    $graph->SetBox(false);

    $graph->img->SetAntiAliasing();

    $graph->yaxis->HideZeroLabel();
    $graph->yaxis->HideLine(false);
    $graph->yaxis->HideTicks(false,false);

    $graph->xgrid->Show();
    $graph->xgrid->SetLineStyle("solid");
    $graph->xaxis->SetTickLabels($datax);
    // $graph->xaxis-
>SetTickLabels(array('9:15','9:30','9:45','10:00'))
    $graph->xgrid->SetColor('#E3E3E3');
    // Create the first line
    // $p1 = new LinePlot(array('90','80','85','55'))
    $p1 = new LinePlot($datay);
    $graph->Add($p1);
    $p1->SetColor("#6495ED");
    $p1->SetLegend('Line 1');

    $graph->legend->SetFrameWeight(1);
    // Output line
    $graph->Stroke();

?>

```

**styles.css file**

```
html
{ height: 100%;}

*
{ margin: 0;
  padding: 0;}

body
{ font: normal .80em 'trebuchet ms', arial, sans-serif;
  background: #F0EFE2;
  color: #777;}

p
{ padding: 0 0 20px 0;
  line-height: 1.7em;}

img
{ border: 0;}

h1, h2, h3, h4, h5, h6
{ font: normal 175% 'century gothic', arial, sans-serif;
  color: #43423F;
  margin: 0 0 15px 0;
  padding: 15px 0 5px 0;}

h2
{ font: normal 175% 'century gothic', arial, sans-serif;
  color: #A4AA04;}

h4, h5, h6
{ margin: 0;
  padding: 0 0 5px 0;
  font: normal 120% arial, sans-serif;
  color: #A4AA04;}

h5, h6
{ font: italic 95% arial, sans-serif;
  padding: 0 0 15px 0;
  color: #000;}

h6
{ color: #362C20;}

a, a:hover
{ outline: none;
  text-decoration: underline;
  color: #1293EE;}

a:hover
{ text-decoration: none;}

.left
{ float: left;
  width: auto;}
```

```
margin-right: 10px;}

.right
{ float: right;
width: auto;
margin-left: 10px;}

.center
{ display: block;
text-align: center;
margin: 20px auto;}

blockquote
{ margin: 20px 0;
padding: 10px 20px 0 20px;
border: 1px solid #E5E5DB;
background: #FFF;}

ul
{ margin: 2px 0 22px 17px;}

ul li
{ list-style-type: circle;
margin: 0 0 6px 0;
padding: 0 0 4px 5px;}

ol
{ margin: 8px 0 22px 20px;}

ol li
{ margin: 0 0 11px 0;}

#main, #logo, #menubar, #site_content, #footer
{ margin-left: auto;
margin-right: auto;}

#header
{ background: #025587;
height: 240px;}

#logo
{ width: 825px;
position: relative;
height: 168px;
background: #025587 url(logo.png) no-repeat;}

#logo #logo_text
{ position: absolute;
top: 20px;
left: 0;}

#logo h1, #logo h2
{ font: normal 300% 'century gothic', arial, sans-serif;
border-bottom: 0;
text-transform: none;
margin: 0;}

#logo_text h1, #logo_text h1 a, #logo_text h1 a:hover
{ padding: 22px 0 0 0;
```

```
    color: #FFF;
    letter-spacing: 0.1em;
    text-decoration: none;}

#logo_text h1 a .logo_colour
{ color: #80FFFF;}

#logo_text h2
{ font-size: 100%;
  padding: 4px 0 0 0;
  color: #DDD;}

#menubar
{ width: 1240px;
  height: 72px;
  padding: 0;
  background: #29415D url(menu.png) repeat-x;}

ul#menu, ul#menu li
{ float: left;
  margin: 0;
  padding: 0;}

ul#menu li
{ list-style: none;}

ul#menu li a
{ letter-spacing: 0.1em;
  font: normal 100% 'lucida sans unicode', arial, sans-serif;
  display: block;
  float: left;
  height: 37px;
  padding: 29px 26px 6px 26px;
  text-align: center;
  color: #FFF;
  text-transform: uppercase;
  text-decoration: none;
  background: transparent;}

ul#menu li a:hover, ul#menu li.selected a, ul#menu li.selected a:hover
{ color: #FFF;
  background: #1C2C3E url(menu_select.png) repeat-x;}

#site_content
{ width: 1180px;
  overflow: hidden;
  margin: 0 auto 0 auto;
  padding: 20px 20px 20px 40px;
  background: #FFF url(content.png) repeat-y;}

#site_content_v2
{ width: 1180px;
  overflow: hidden;
  margin: 0 auto 0 auto;
  padding: 20px 20px 20px 40px;
  background: #FFF url(content_2.png) repeat-y;}

.sidebar
{ float: right;
```

```
width: 280px;
padding: 0 15px 20px 15px;}

.sidebar ul
{ width: 178px;
padding: 4px 0 0 0;
margin: 4px 0 30px 0;}

.sidebar li
{ list-style: none;
padding: 0 0 7px 0; }

.sidebar li a, .sidebar li a:hover
{ padding: 0 0 0 40px;
display: block;
background: transparent url(link.png) no-repeat left center;}

.sidebar li a.selected
{ color: #444;
text-decoration: none;}

#content
{ text-align: left;
width: 800px;
padding: 0;}

#content_v2
{ text-align: left;
width: 1180px;
padding: 0;}

#content ul
{ margin: 2px 0 22px 0px;}

#content ul li
{ list-style-type: none;
background: url(bullet.png) no-repeat;
margin: 0 0 6px 0;
padding: 0 0 4px 25px;
line-height: 1.5em;}

#footer
{ width: 1240px;
font: normal 100% 'lucida sans unicode', arial, sans-serif;
height: 33px;
padding: 24px 0 5px 0;
text-align: center;
background: #29425E url(footer.png) repeat-x;
color: #FFF;
text-transform: uppercase;
letter-spacing: 0.1em;}

#footer a
{ color: #FFF;
text-decoration: none;}

#footer a:hover
{ color: #FFF;
text-decoration: underline;}
```

```
.search
{ color: #5D5D5D;
  border: 1px solid #BBB;
  width: 134px;
  padding: 4px;
  font: 100% arial, sans-serif;}

.form_settings
{ margin: 15px 0 0 0;}

.form_settings p
{ padding: 0 0 4px 0;}

.form_settings span
{ float: left;
  width: 200px;
  text-align: left;}

.form_settings input, .form_settings textarea
{ padding: 5px;
  width: 299px;
  font: 100% arial;
  border: 1px solid #E5E5DB;
  background: #FFF;
  color: #47433F;}

.form_settings .submit
{ font: 100% arial;
  border: 1px solid;
  width: 99px;
  margin: 0 0 0 10px;
  height: 33px;
  padding: 2px 0 3px 0;
  cursor: pointer;
  background: #263C56;
  color: #FFF;}

.form_settings textarea, .form_settings select
{ font: 100% arial;
  width: 299px;}

.form_settings select
{ width: 310px;
  height: 33px;
  clear: left;
}

.form_settings .checkbox
{ margin: 4px 0;
  padding: 0;
  width: 14px;
  border: 0;
  background: none;}

.separator
{ width: 100%;
  height: 0;
  border-top: 1px solid #D9D5CF;}
```

```
border-bottom: 1px solid #FFF;  
margin: 0 0 20px 0;}
```

```
table  
{ margin: 10px 0 30px 0;}
```

```
table tr th, table tr td  
{ background: #3B3B3B;  
color: #FFF;  
padding: 7px 4px;  
text-align: left;}
```

```
table tr td  
{ background: #F0EFE2;  
color: #47433F;  
border-top: 1px solid #FFF;}
```